Functional and Reactive programming using Facebook's React and Appian's SAIL

January 12th, 2016

Carlos Aguayo, Software Engineer Manager

Appian

Welcome!

CARLOS
AGUAYO

SOFTWARE ENGINEER MANAGER
@carlosaguayo81

Overview

At this lecture, you will:

- Learn about programming paradigms for building modern UIs
  - Functional Programming
  - Reactive Programming

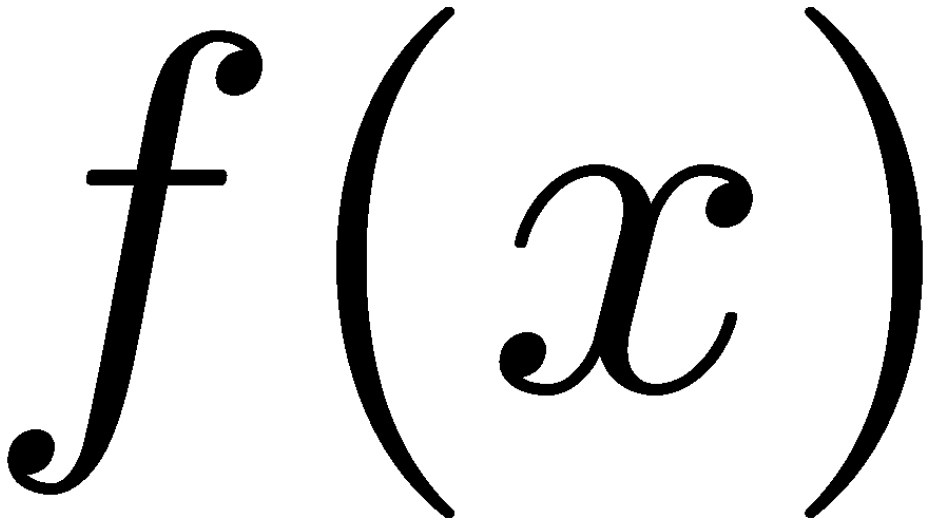
- See an example of how Appian applies these techniques

Definitions

Functional Programming: a style of programming that emphasizes the use of functions as first-class objects and avoids mutable state and side-effects

Reactive Programming: a style of programming oriented around data flow and propagation of changes triggered by events (think UI interactions)

Functional Programming



Functional Programming

- **No mutable state**
  - e.g. no arrays, no variables, no side-effects
  - safe to share state
  - free to cache state
  - no concurrency problems
  - parallelizable

Functional Programming

- **First-class and higher-order functions** instead of explicit control flow
  - e.g. No while or for loops

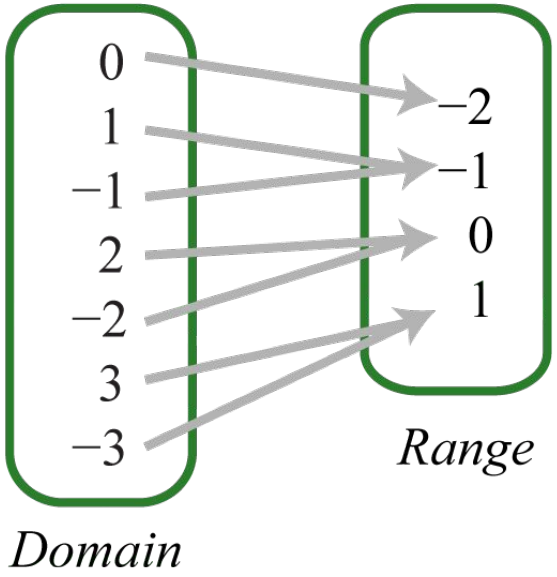
```
var information;

if(userInSuperSecureGroup(loggedInUser()) {
  information = superSecureInformation();
} else {
  information = boringDisplay();
}
```

```
var information = if(
  userInSuperSecureGroup(loggedInUser()),
  superSecureInformation(),
  boringInformation()
);
```

Functional Programming

- **Referentially transparent** functions (pure functions)
  - Returns same output each time for a given input

$$|x| - 2 =$$



How do we handle user interactions?

Event-Driven: Events trigger blocks of imperative code. The programmer must **explicitly propagate state change** throughout the UI

Reactive: Events trigger programmer-defined state change, which is **implicitly propagated** throughout the UI

FRP: Functional Reactive Programming

FRP is about declarative programming with **time-varying values** using functional building blocks

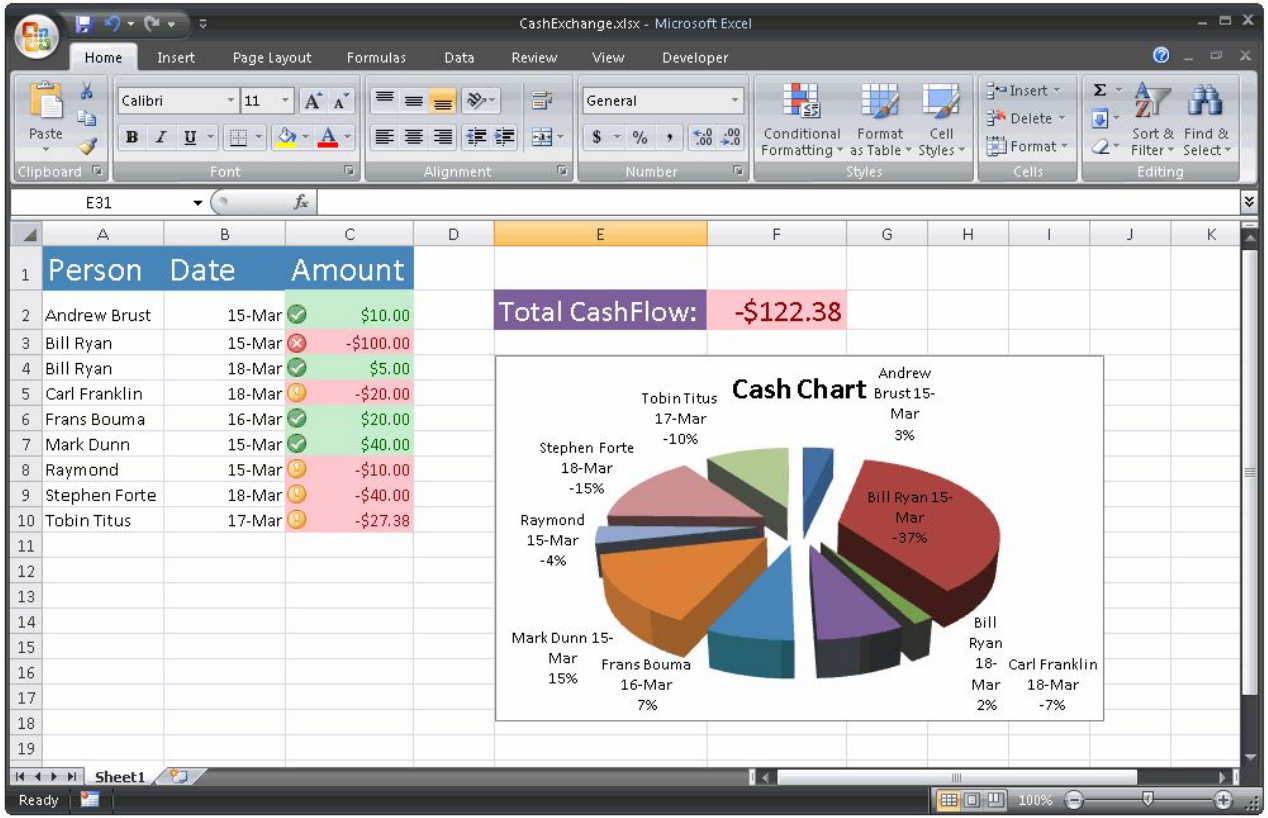
FRP: Functional Reactive Programming

In FRP, **time is implicit** as the code describes a UI at any moment in time.

FRP: Functional Reactive Programming

FRP provides declarative control flow structures for events. There are **no event handlers** or **callbacks**

Spreadsheets: Most popular FRP systems



Facebook's React

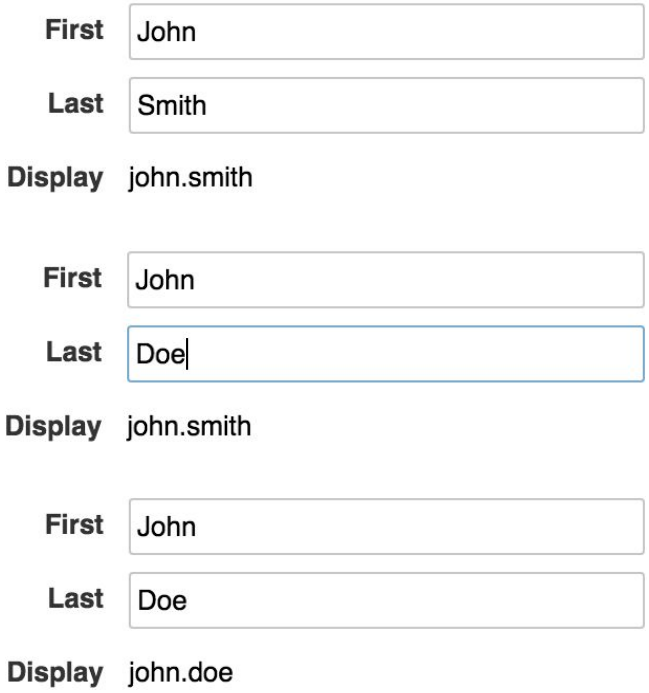
Code snippets based from: http://tylermcginnis.com/reactjs-tutorial-a-comprehensive-guide-to-building-apps-with-react/

Functional Reactive Programming at Appian

Appian SAIL: Functional Reactive Programming

```
load(
  first: "John",
  last: "Smith",
  with(
    display: lower(concat(first, ".", last))
    {
      a!textField(label: "First", value: first, saveInto: first),
      a!textField(label: "Last", value: last, saveInto: last),
      a!textField(label: "Display", value: display, readOnly: true)
    }
  )
)
```

Appian SAIL: Functional Reactive Programming

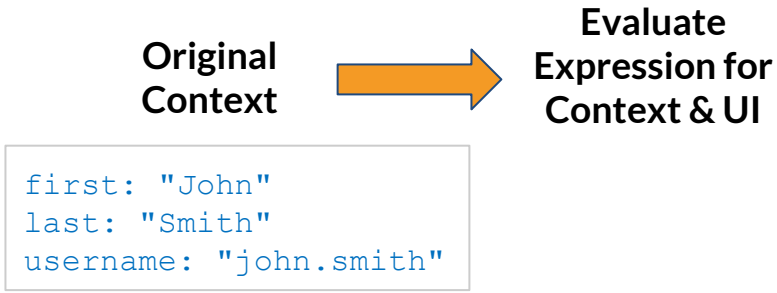



```
first: "John",
last: "Smith",
display: "john.smith"
```

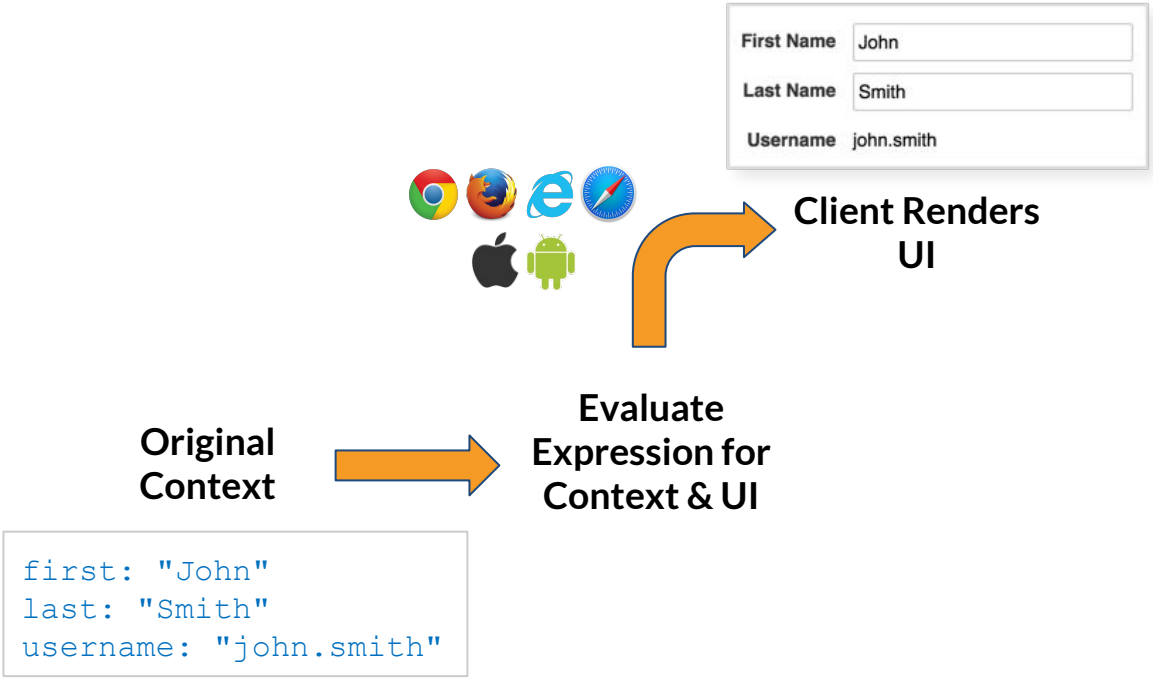
```
first: "John",
last: "Doe",
display: "john.smith"
```

```
first: "John",
last: "Doe",
display: "john.doe"
```

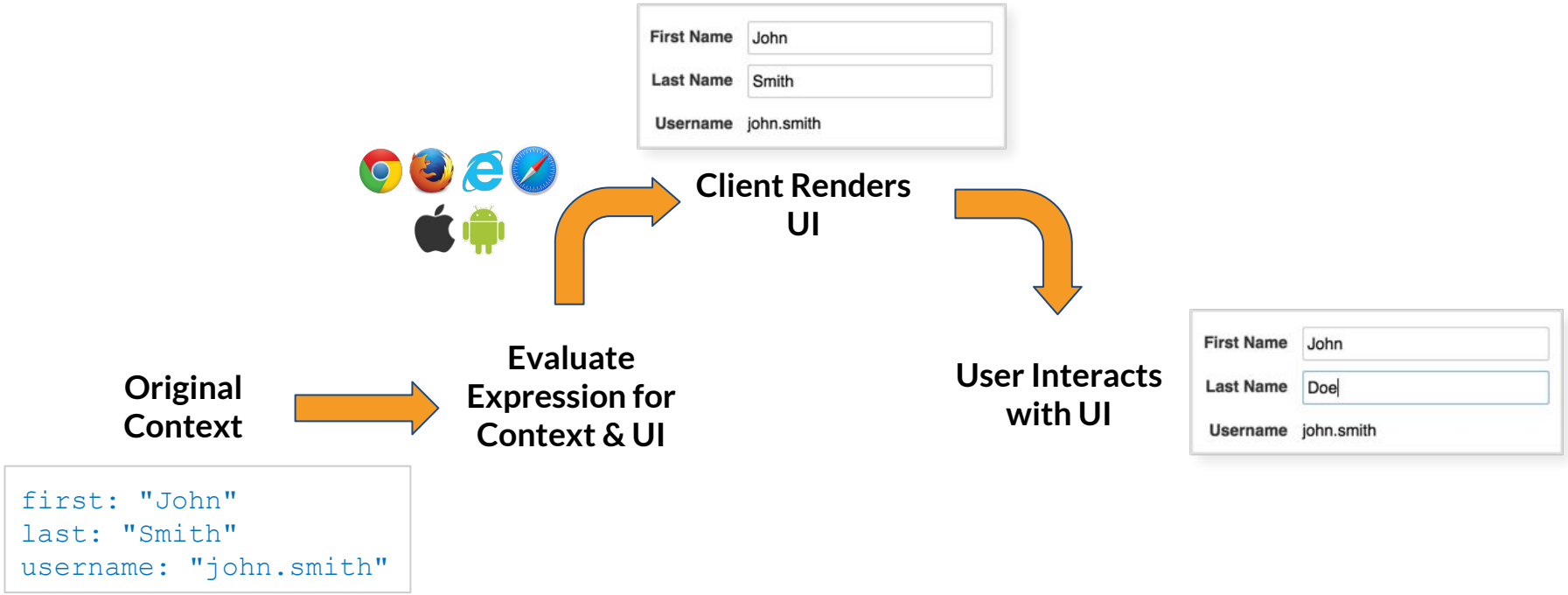
Appian SAIL: Functional Reactive Programming



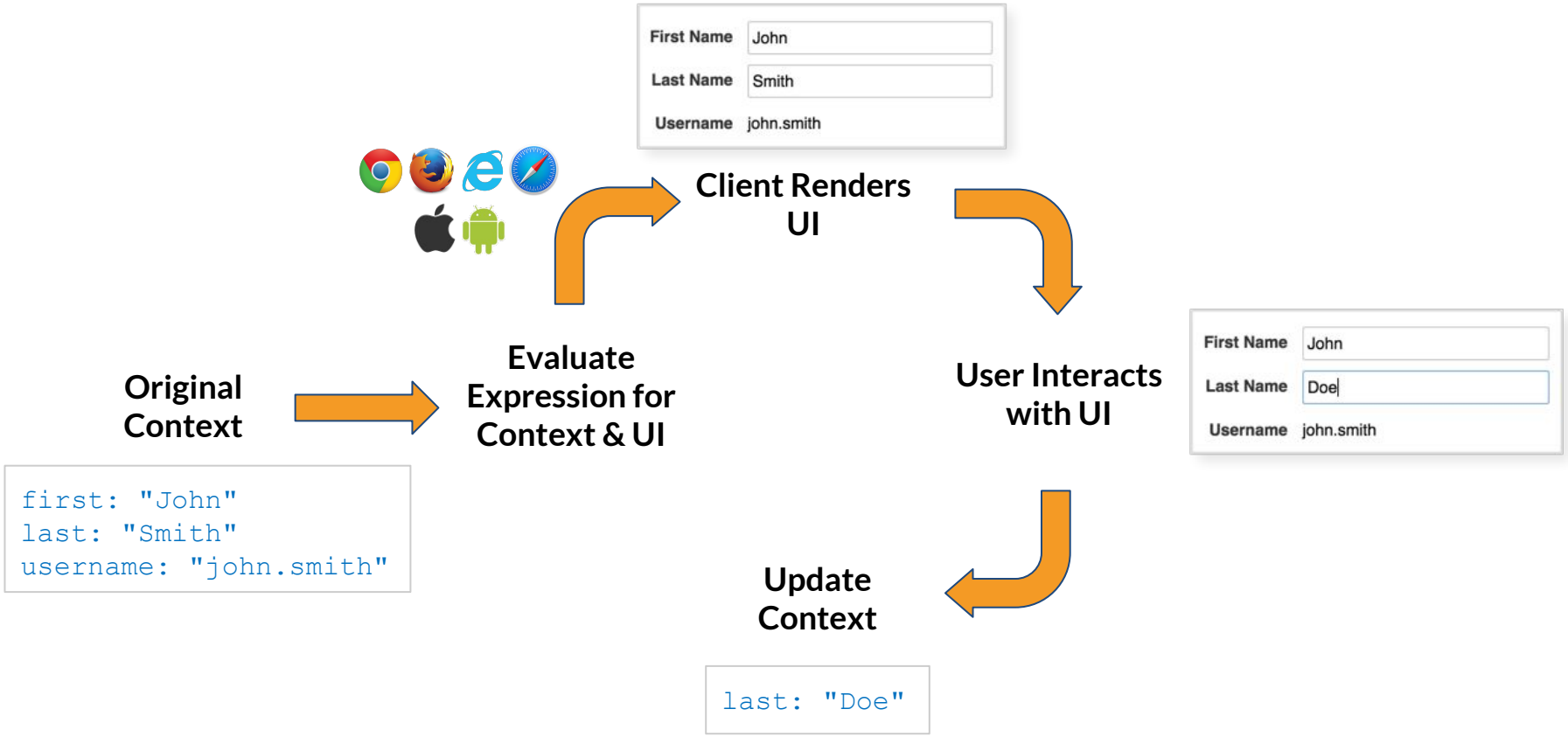
Appian SAIL: Functional Reactive Programming



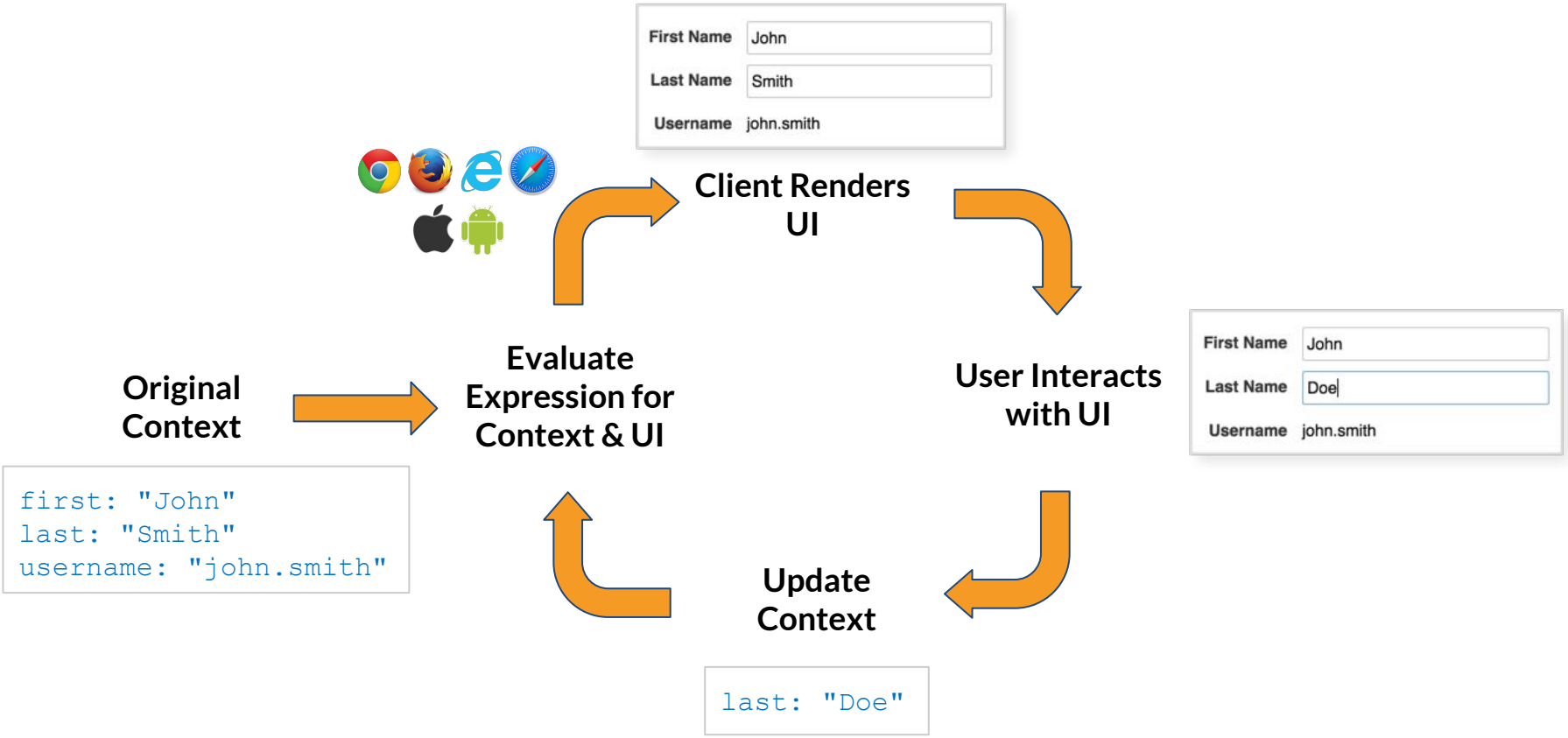
Appian SAIL: Functional Reactive Programming



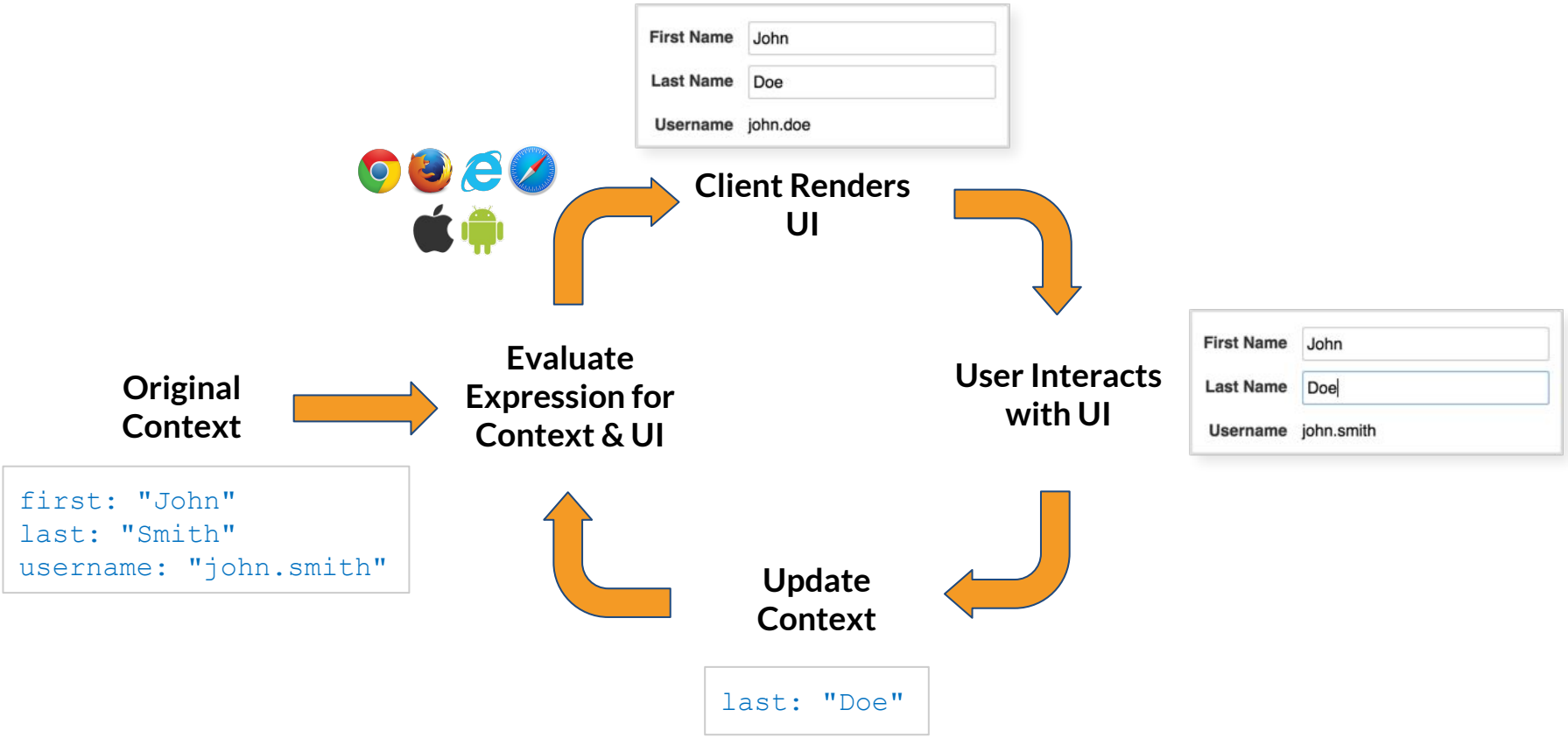
Appian SAIL: Functional Reactive Programming



Appian SAIL: Functional Reactive Programming



Appian SAIL: Functional Reactive Programming



SAIL: Easy to Use and Easy to Build!

- ☑ User interaction driven
- ☑ Integration with external sources
- ☑ Cross-platform support for all UIs

Why Functional Reactive UIs?

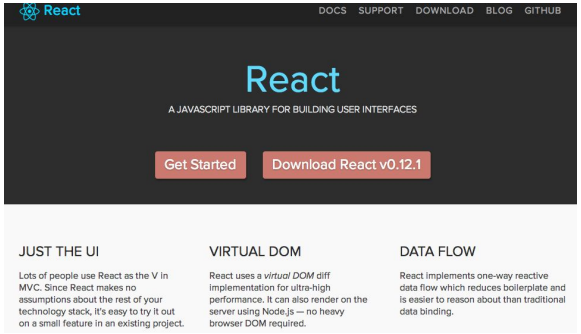
- **Easier to test & reason about**
  - Given some data (inputs), render a UI
  - Function describes state at any given point in time
  - UI functions only depend on inputs, not hidden state, not global state. i.e. stateless UIs
- **Composable**
  - UIs are assembled by declaratively gluing functions
- **Powerful**
  - Functions encapsulate business and UI logic
  - No separate, underpower, templating language
  - Templates are functions, placeholders are parameters

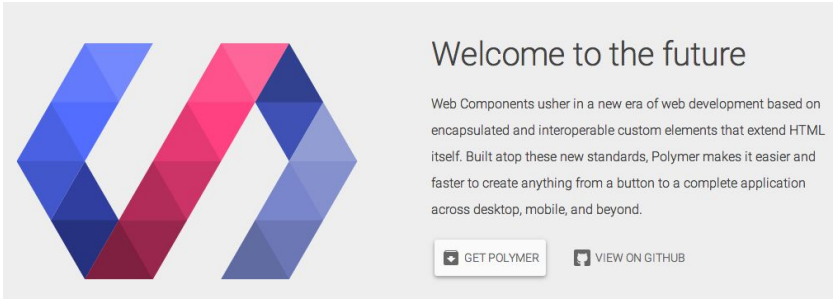
Why a One-Way Data Flow?

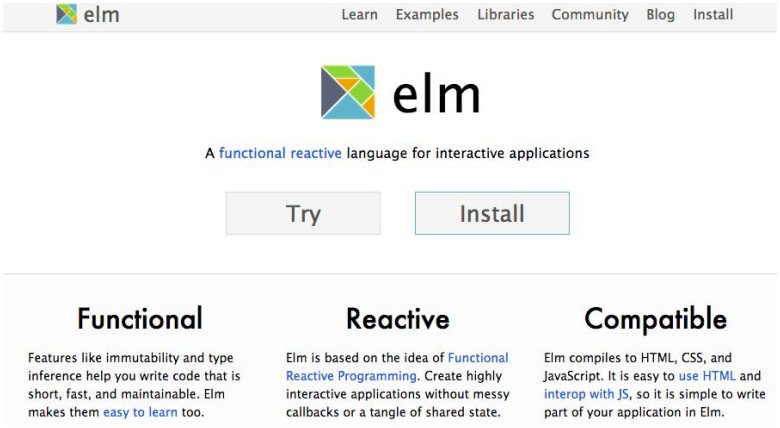
- **No cascading effects**
  - Views do not directly modify the model. They “ask” the FRP runtime to do it.
  - No dirty checking, no infinite loops checks
- **Single channel for state changes**
  - Good for Debugging
  - Good for Logging
- **All side-effects are controlled by the framework**
  - Allows Recording
  - Allows Replaying, etc

Relevant Technologies

- React.js / Om
  - http://facebook.github.io/react/
  - https://github.com/swannodette/om
- Elm
  - http://elm-lang.org/
- Polymer
  - https://www.polymer-project.org/







Relevant Research

- **Functional Reactive Animation**

  http://conal.net/papers/icfp97/icfp97.pdf
- **Asynchronous Functional Reactive Programming for GUIs**
  - http://people.seas.harvard.edu/~chong/pubs/pldi13-elm.pdf
- **Composable, Demand-Driven Incremental Computation**
  - http://www.cs.umd.edu/~jfoster/papers/cs-tr-5027.pdf

Appian Engineering & Internship

http://www.appian.com/about/careers/life-at-appian/

http://www.appian.com/about/careers/university-recruiting/

http://www.appian.com/about/careers/engineering/


thank you!