React in the real world

A short talk about data flow
«the heart of any React app»

Sven A Robbestad

Frontend developer at Inmeta Consulting AS

Inmeta

Privat Bedrift Nettbutikk Min bedrift Min side Trådløs bedrift Ola Olaisen

Oversikt

Søk...

Antall abonnenter	Aktive	Sperrede	Reserverte
0	0	0	0

Detaljer

Organisasjonsnavn	ABS-PARKETTGRUPPEN NORGE AS
Organisasjonsnummer	976 390 512
Addresse	
Avtaletype	
Avtaledato	

Kontakt

Kontakt NetCom dersom innholdet er feil.

Legg til abonnent

Legg til fakturasted

Administrer grupper

Bestill SIM-kort

Inmeta

Privat Bedrift Nettbutikk Min bedrift Min side Trådløs bedrift Ola Olaisen

Rapporter Søk...

Totalforbruk

Totalforbruk, per abonnent	Siste fakturaperiode
Totalforbruk, per fakturasted	Siste fakturaperiode
Totalforbruk, per gruppe	Siste fakturaperiode

Tjenester og varekjøp

Tjenester og varekjøp, per abonnent	Siste fakturaperiode
Tjenester og varekjøp, per fakturasted	Siste fakturaperiode
Tjenester og varekjøp, per gruppe	Siste fakturaperiode

Dataforbruk

Dataforbruk, per abonnent	Siste fakturaperiode
Dataforbruk, per fakturasted	Siste fakturaperiode
Dataforbruk, per gruppe	Siste fakturaperiode

Inmeta

Endre status

Sperre abonnement

Lukk

Abonnementet sperres fordi

- (●) Mobilen er stjålet
- ( ) Mobilen er mistet

Det kan ta opptil 10 minutter før dette oppdateres

Neste

Data flow

One of the major reasons for using React is the Flux architecture.

The idea that you have to use MVC to create web apps was discarded by the React team, and that has been an eye opener.

Data flow

As a consequence there are a lot of Flux implementations in the wild:

Fluxxor, Reflux, McFly, Marty, Fluxible, Fynx, DeLorean

And the list goes on

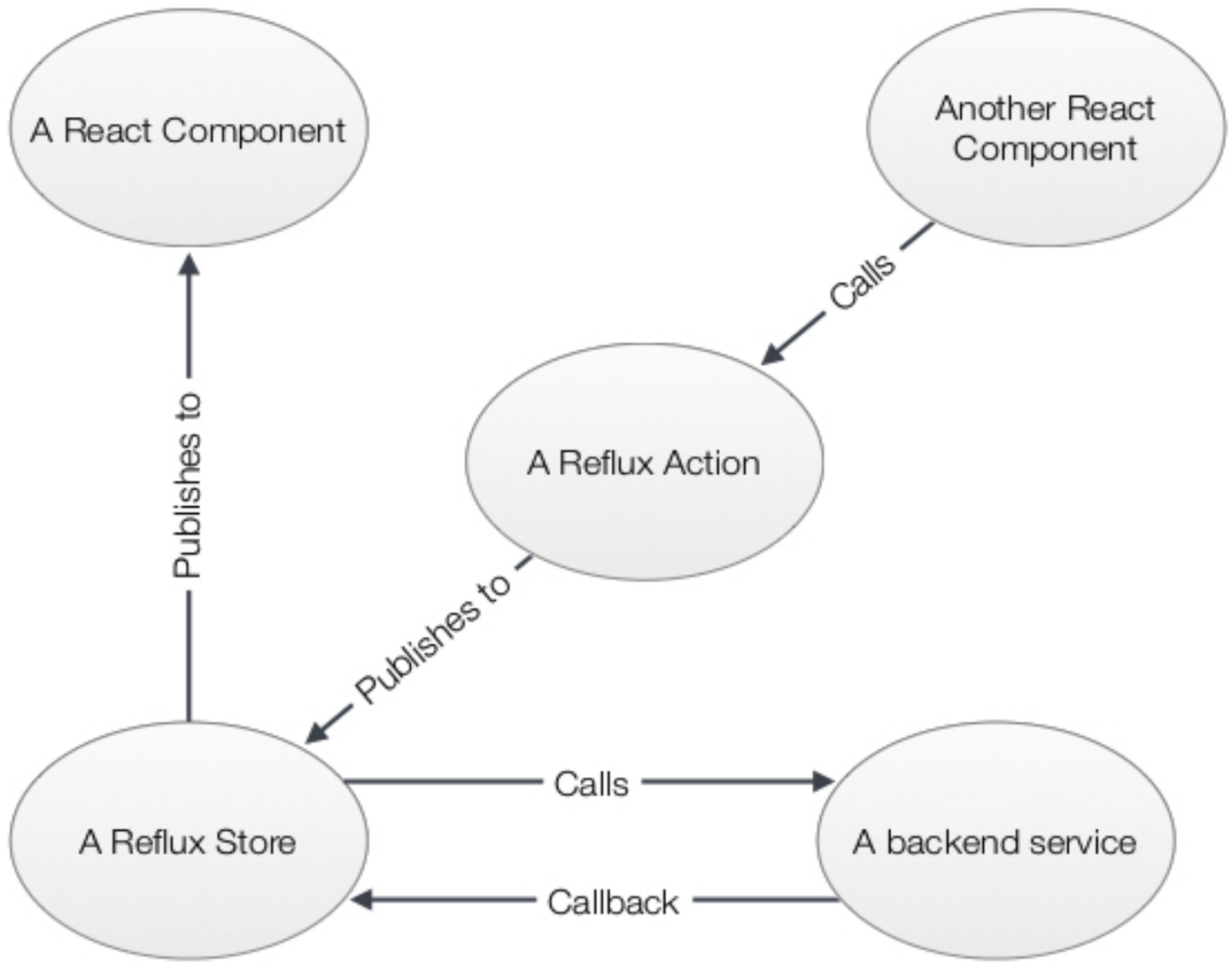
Data flow

After much deliberation, we landed on Reflux

The fundamental reasion is that the implementation was easy to understand and easy to implement

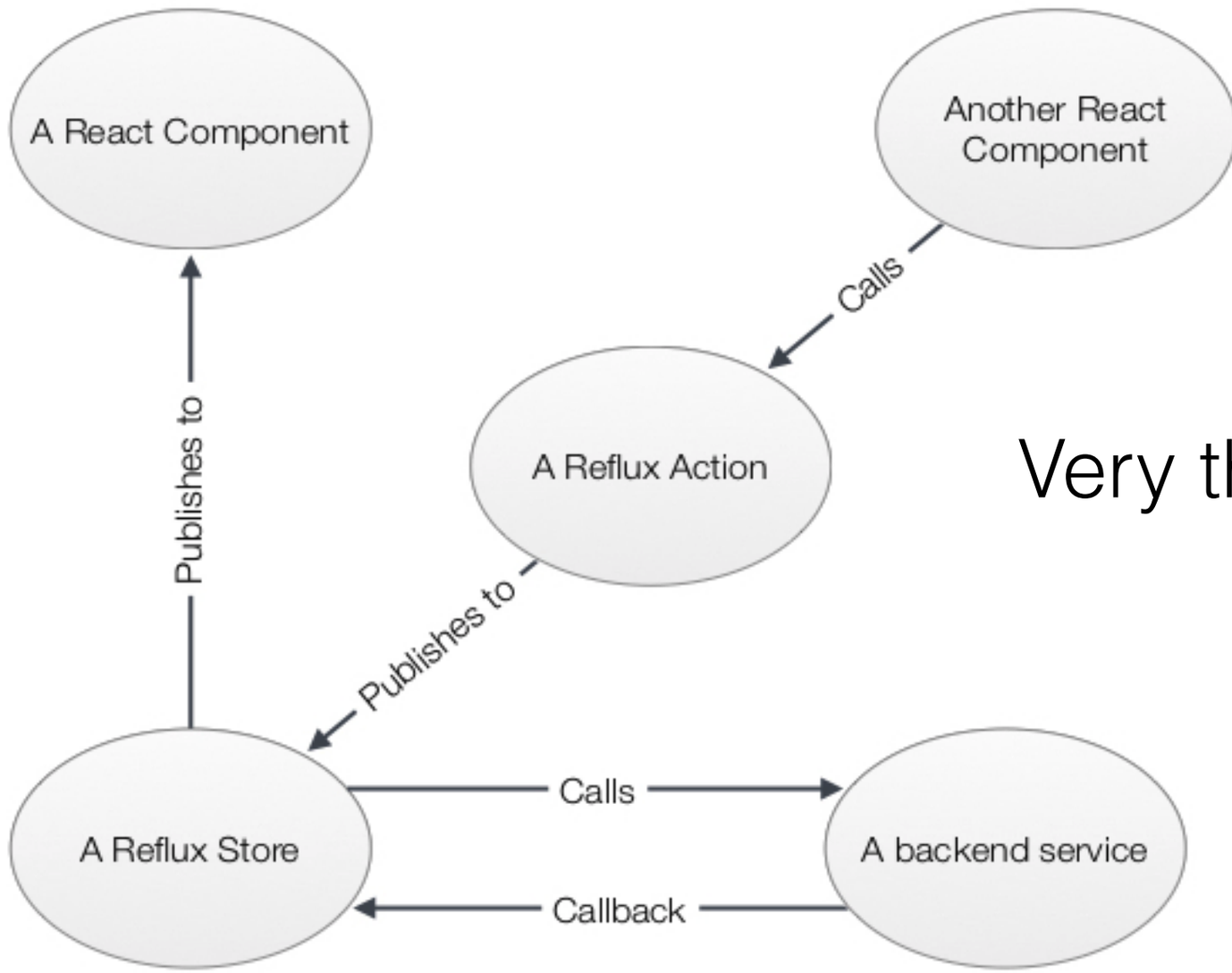
However, after a while we decided to use it in a slightly different way than it's described in the docs. In the following slides I'll share some of our experiences with Reflux.

Regular Reflux



Regular Reflux

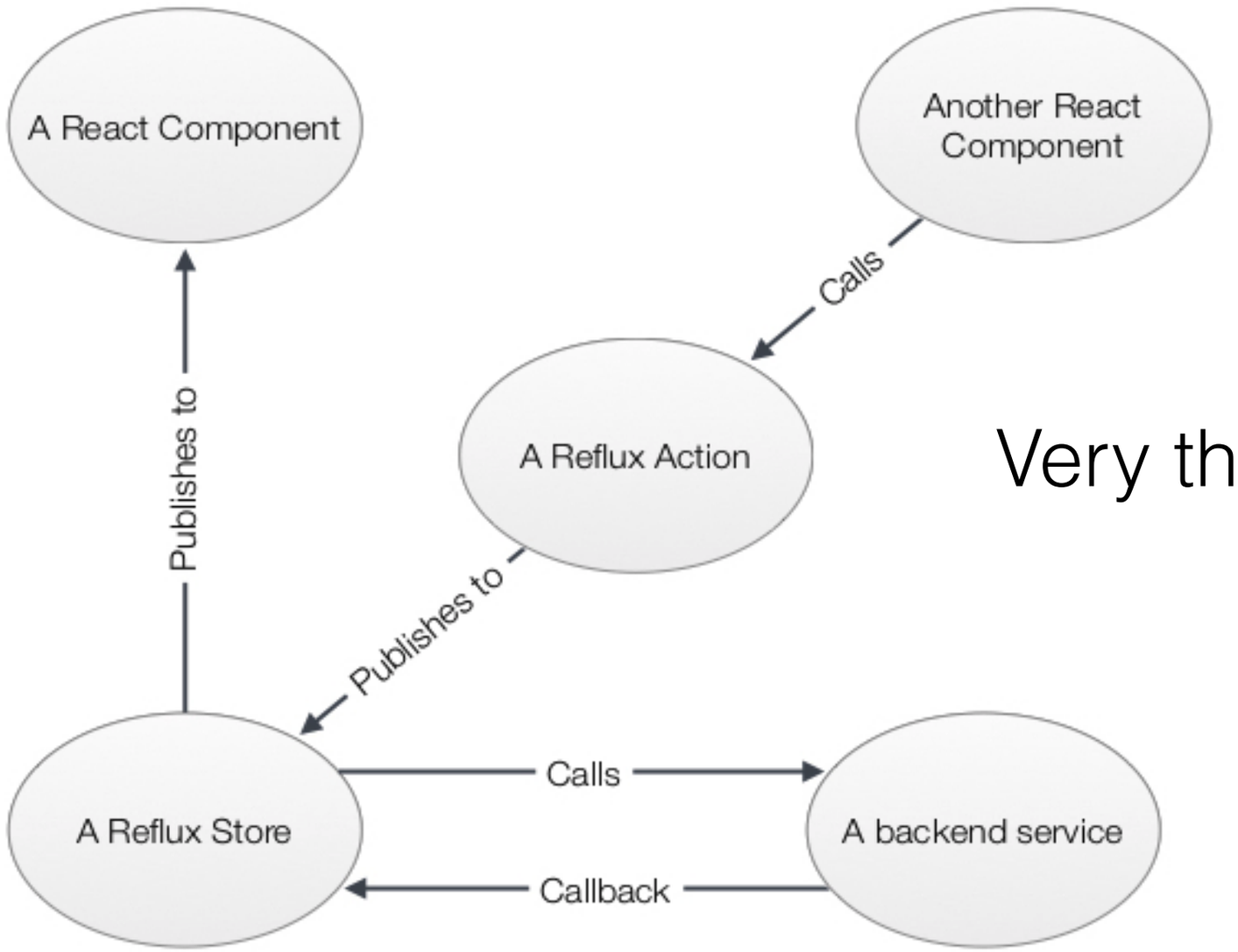
Inmeta



Very thin actions

Regular Reflux

Inmeta

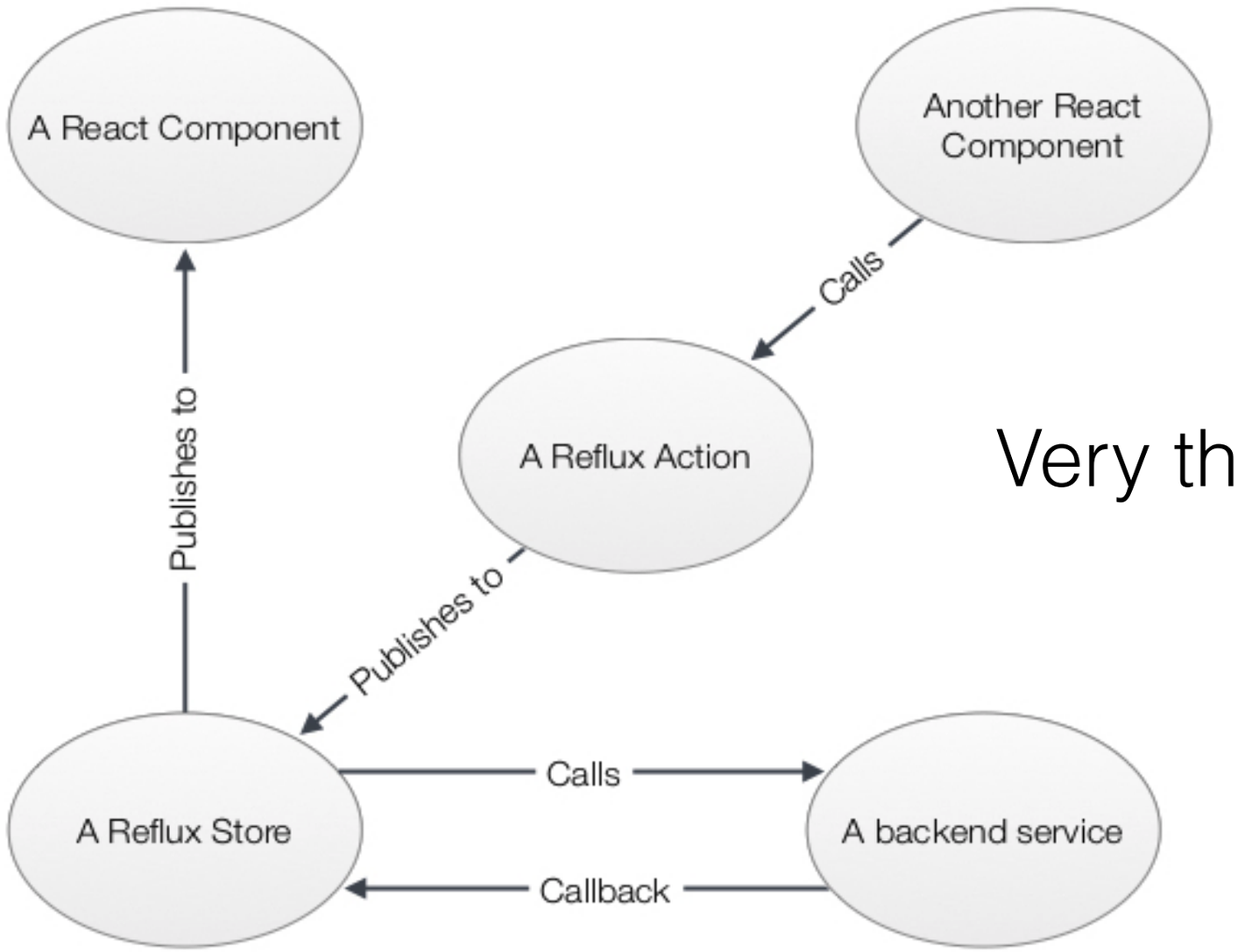


Very thin actions

Very fat stores

Regular Reflux

Inmeta



Very thin actions

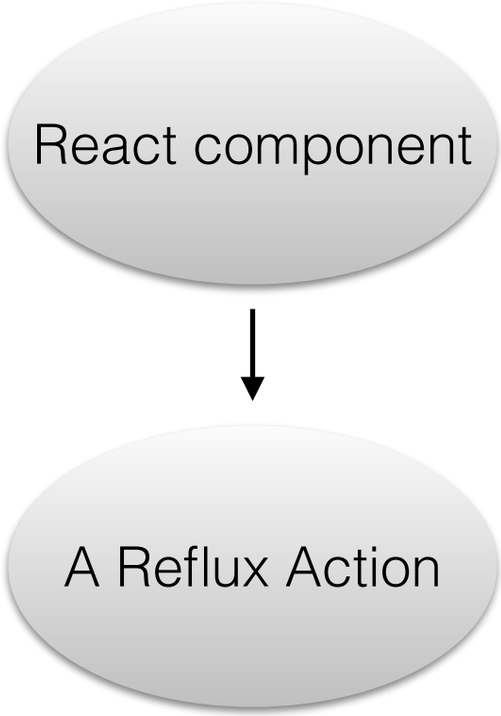
Very fat stores

Services called from stores
can get hard to debug
Also hard to test

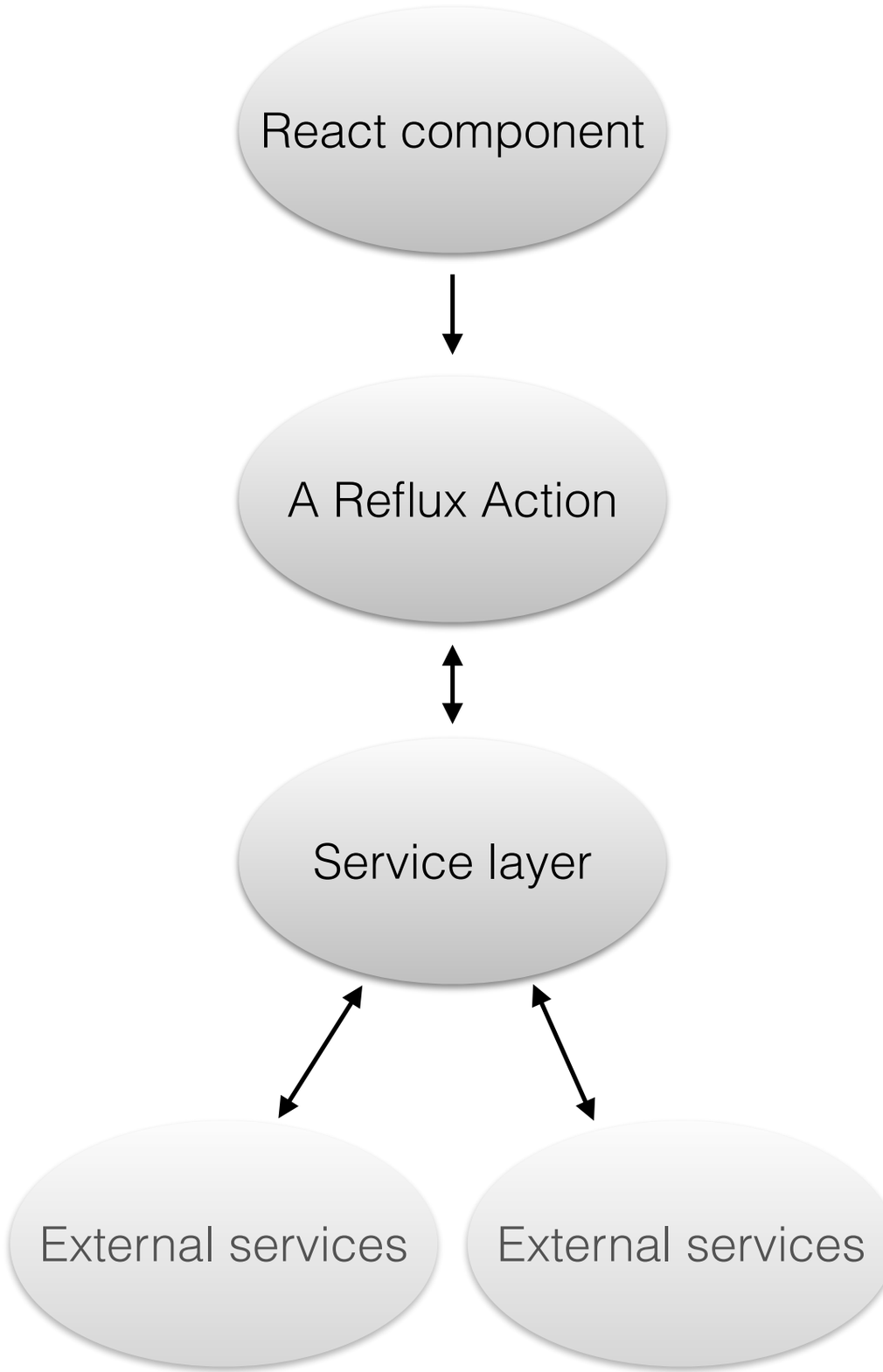
How this looks

```
// Creating an Action
var makeRequest = Reflux.createAction();


// A Reflux Store
var statusStore = Reflux.createStore({
    init: function() {
        this.listenTo(makeRequest, this.onMakeRequest);
    },
    onMakeRequest: function(url) {
    // Assume `request` is some HTTP library (e.g. superagent)
        request(url, function(response) {
            if (response.ok) {
                makeRequest.completed(response.body);
            } else {
                makeRequest.failed(response.error);
            }
        })
    }
});


// A simple view component that outputs to console
function ConsoleComponent() {
    statusStore.listen(function(data) {
        console.log('data: ', ata);
    });
}
```

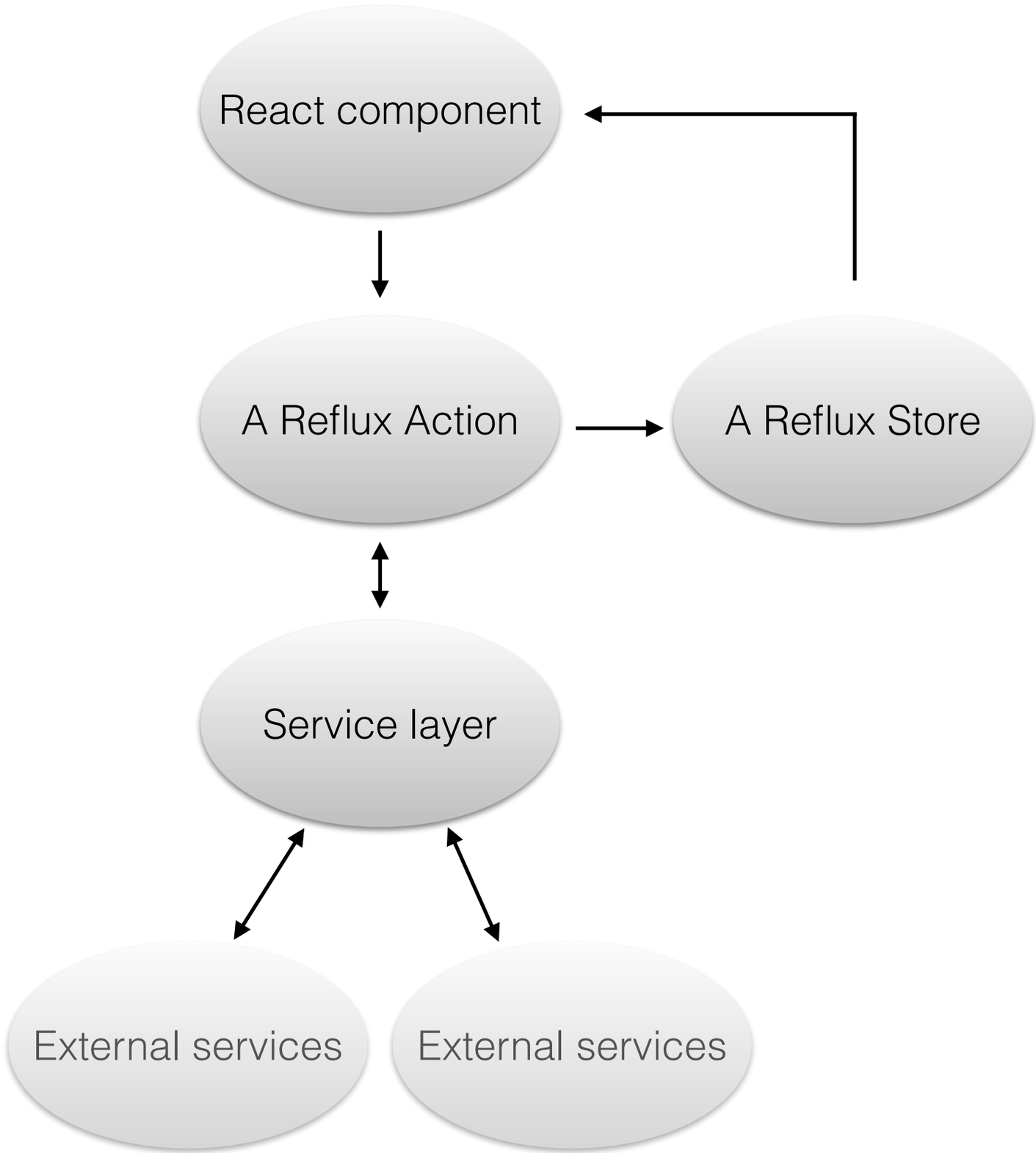
Improved Reflux



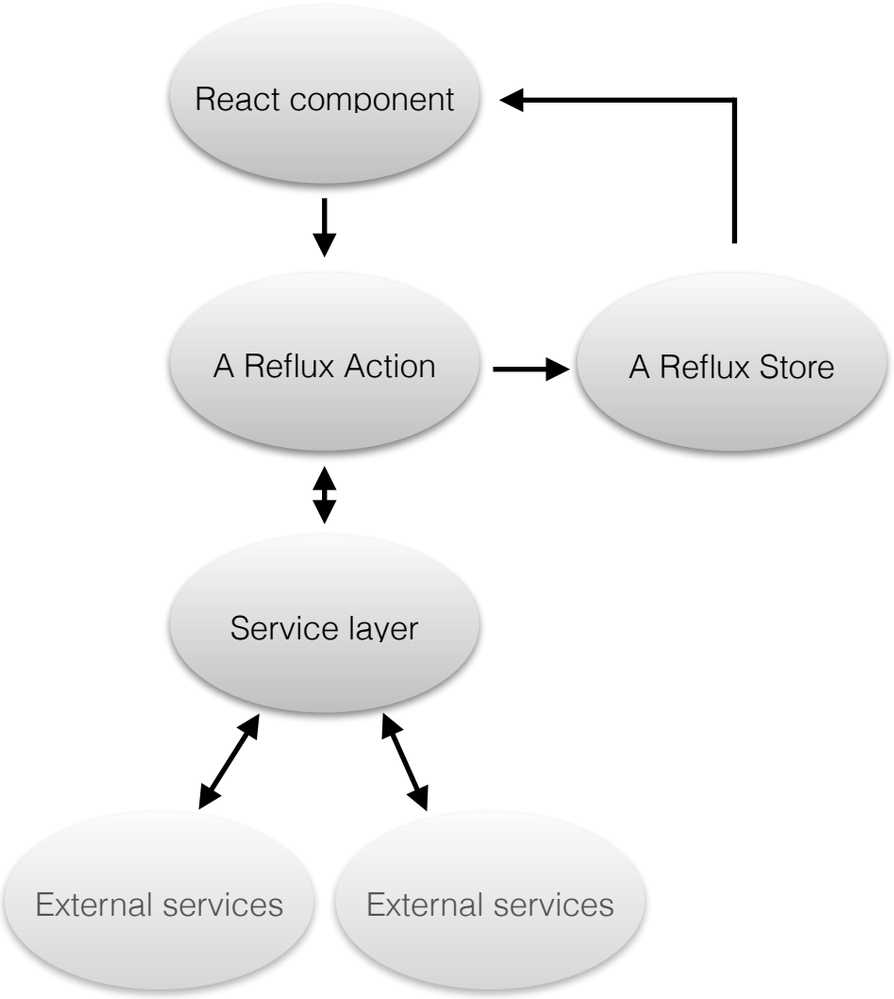
Improved Reflux



Improved Reflux



Improved Reflux



Benefits

Better separation of concerns

- Easier to test

Easier to structure data flow

Unnecessary to change Reflux, it's just a better way of using it. Reflux already supports this architecture

How this looks

```
var actions = {
    getRequestData: Reflux.createAction("getDataFromServer"),
    getRequestDataSuccess: Reflux.createAction('getDataFromServerSuccess')
};
actions.getDataFromServer.listen(function (url) {
    DataService.getData(url).
        then(function (response) {
            actions.getRequestDataSuccess(data);
        }).catch(function (err) {
           // do some error handling
        });
});

class DataService {
    getData(url){
        var req = request.get(url)
            .set('Content-Type', this.contentType)
        return new Promise(function (resolve, reject) {
            req.end(function (err, res) {
                resolve(res.text);
            });
        });
    }
}
```

```
const DataStore = Reflux.createStore({
    init(){
        this.listenTo(actions.getDataFromServer, this.getDataFromServer);
    },
    getDataFromServer: function(result){
        this.trigger(result.message);
    }
});

const ConsoleComponent = React.createClass({
    mixins: [
        Reflux.listenTo(actions.getDataFromServerSuccess, "getData")
    ],
    getData(){
        console.log(data);
    }
    render(){
        return <div />
    }
}
```