







- Display always fresh data
- Do only little extra coding



Firestore

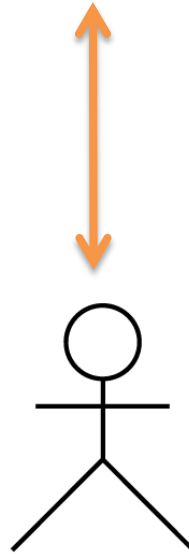


Not everything is awesome.

- Rich Hickey



 **Firebase**



Firestore Database

Dashboard VIEWING WORDYTEST






- Data
 - order
 - organisation
 - settings
 - team
 - transaction
 - user
 - credentials
 - organisation
 - profile
 - K1YllvLI9Jt8uUHGrzW
 - K1YxlNqqIbGWLth-TJY
 - K1d-3u0bWpl6ytBCRgB
 - K1d9loUr2HltdbaoLbW
 - email: "jsurovcik@centrum.sk"
 - emailNotifications
 - gravatarHash: "9ec3b2b10614a809ed5848b6d01538fb"
 - orgRole: "user"
 - K1dAL069b6ukb6KD6Of
 - email: "hellboy4247+user3@gmail.com"
 - emailNotifications
 - gravatarHash: "3dc7c95e3092b4e34bf0f04f329e01"- Security & Rules
- Simulator
- Analytics
- Login & Auth
- Hosting
- Secrets


```
firebase.child('events/123/city').set('Bratislava')
```

```
firebase.child('events/123/').on('value', function(value) {  
  alert(value)  
})
```

```
firebase.child('events/123/city').set('Bratislava')
```

```
firebase.child('events/123/').on('value', function(value) {  
  alert(value)  
})
```

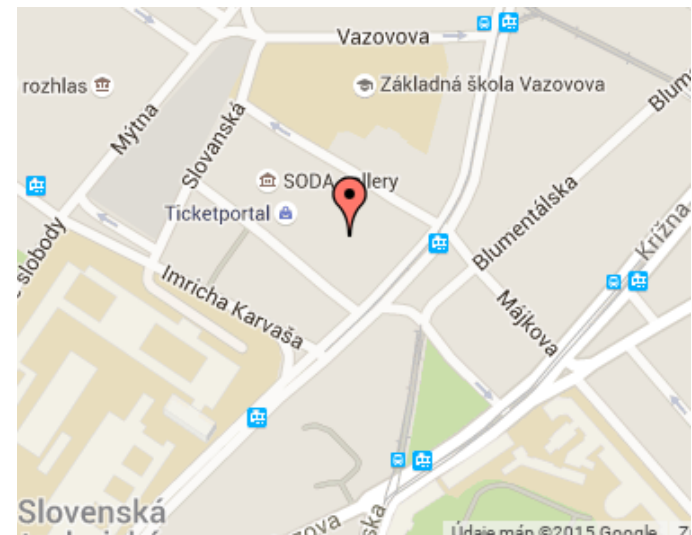
	Emily	Jogging @ Petrzalka
	John	Beer @ Bratislava
	Eve	Cinema @ Vienna
	Ford	Wild party @ Saffron
	Arthur	Tea @ Teahouse

Initiator: Ford Prefect

Description: Party

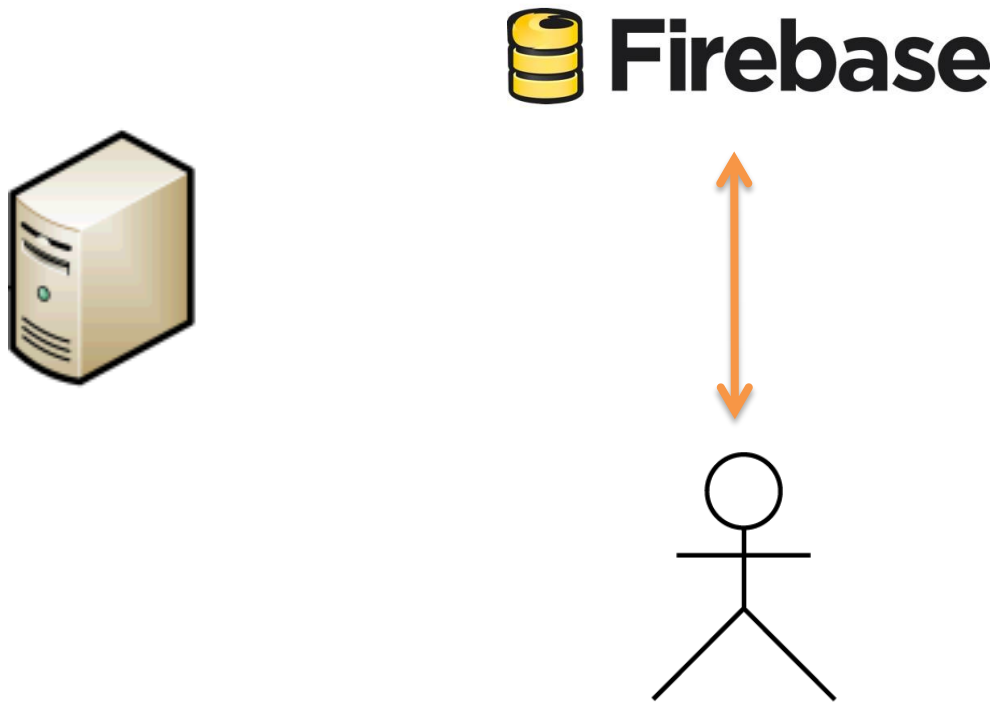
Place: Hotel Saffron, Earth

Remarks: Bring your own towel



```
const ListenEventDetails = listenFirebase(  
  (props) => props.firebase.child('events').child(props.id),  
  (dispatch, props, data) => {  
    dispatch(onDetails, [props.id, data])  
  }  
)
```

```
render() {  
  const event = props.events[props.id]  
  return <div>  
    <ListenEventDetails id = {props.id} />  
    {event !== null ?  
      <div>  
        <input value={event.owner} />  
        <input value={event.description} />  
        ...  
      </div>  
      :  
      <div> 'Loading...' </div>  
    }  
  </div>  
}
```



Eliminating read API endpoints

GET `api.mycoolapp.com/event/{id}`

GET api.mycoolapp.com/event/{id}

GET api.mycoolapp.com/sql/"select *
from users join events on
users.id = events.id where
users.id = 123"

GET api.mycoolapp.com/event/{id}

GET api.mycoolapp.com/sql/"select *
from users join events on
users.id = events.id where
users.id = 123"

GET api.mycoolapp.com/sql/"delete *
from users"

Oops..

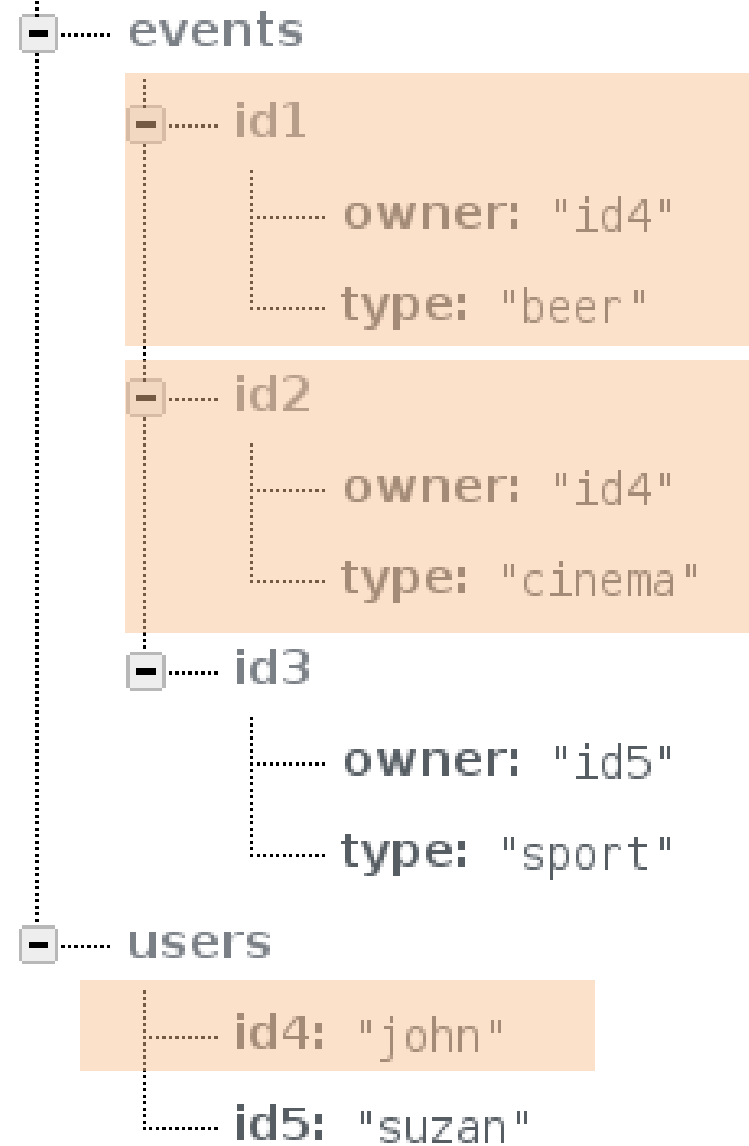
Let

`user/$id`

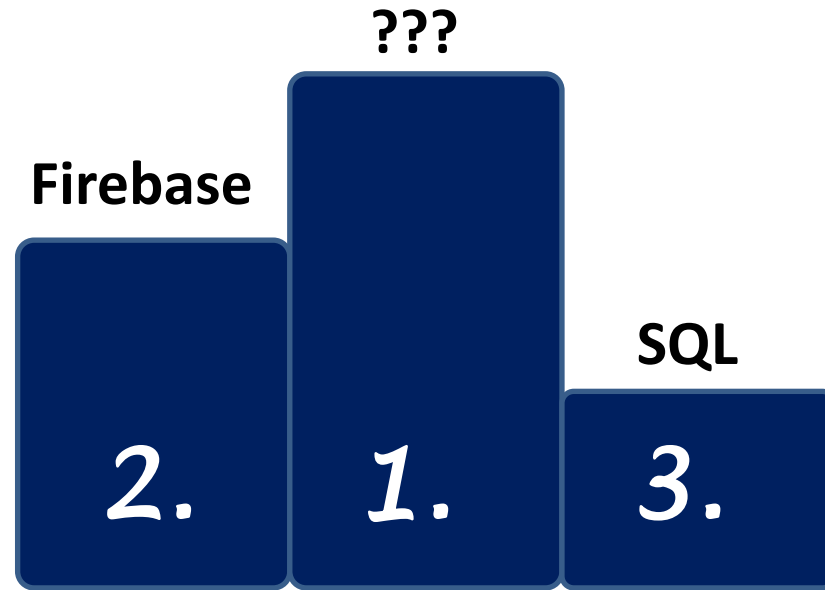
be guarded by

```
function(who, what, DB) {  
  if who == $id and only 'name'  
    changes  
}
```

reactive2015



Query validating capabilities



```
firebase.child('events/123/city').set('Bratislava')
```

```
firebase.child('events/123/city').set('Bratislava')
```

Transactions? Nope.

```
firebase.child('events/123/city').set('Bratislava')
```

Transactions? Nope.

```
{  
  type: 'pay',  
  from: 'john',  
  to: 'emily',  
  ammount: '10 bucks'  
}
```

firebase-transactions



<https://github.com/vacuumlabs/firebase-transactions>

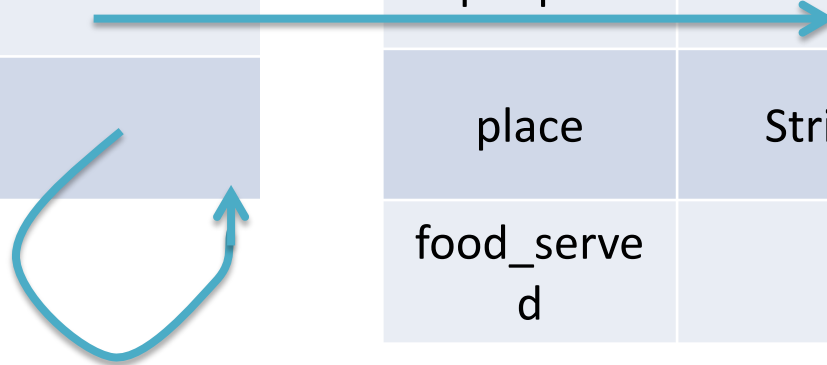
Not everything is JSON document

-- old Chinese proverb --



User	
name	String
attends	
friends	

Event	
type	String
people	
place	String
food_serve d	



etc

```
(d/q '[ :find ?name
       :where [?name :name ?person]
               [?person :attends ?event]
               [?event :food_served ?food]
               [?food :contains ?ingredient]
               [?ingredient :name "chocolate"] ]
```

Reversibility: What subscriptions (and how) are affected by a given change?

change:

```
firebase.child('events/123/place').set(...)
```

```
sub1: firebase.child('events/234').on(...)
```

```
sub2: firebase.child('events/123').on(...)
```

```
sub3: firebase.child('users/567/name').on(...)
```

```
sub4: firebase.child('events').on(...)
```

```
(d/q '[ :find ?name
       :where [?name :name ?person]
               [?person :attends ?event]
               [?event :food_served ?food]
               [?food :contains ?ingredient]
               [?ingredient :name "chocolate"] ]
```

Summary

- Reactivity is great, especially when cheap
- Probably quite doable with right tools
- HARD
- Several subproblems must be tackled at once
- Firebase, GraphQL, Relay, Falcor, Datomic, ...
- Let's get to work

Questions?



tomas.kulich@vacuumlabs.com



@tomas_kulich