

FLUX INTRODUCTION

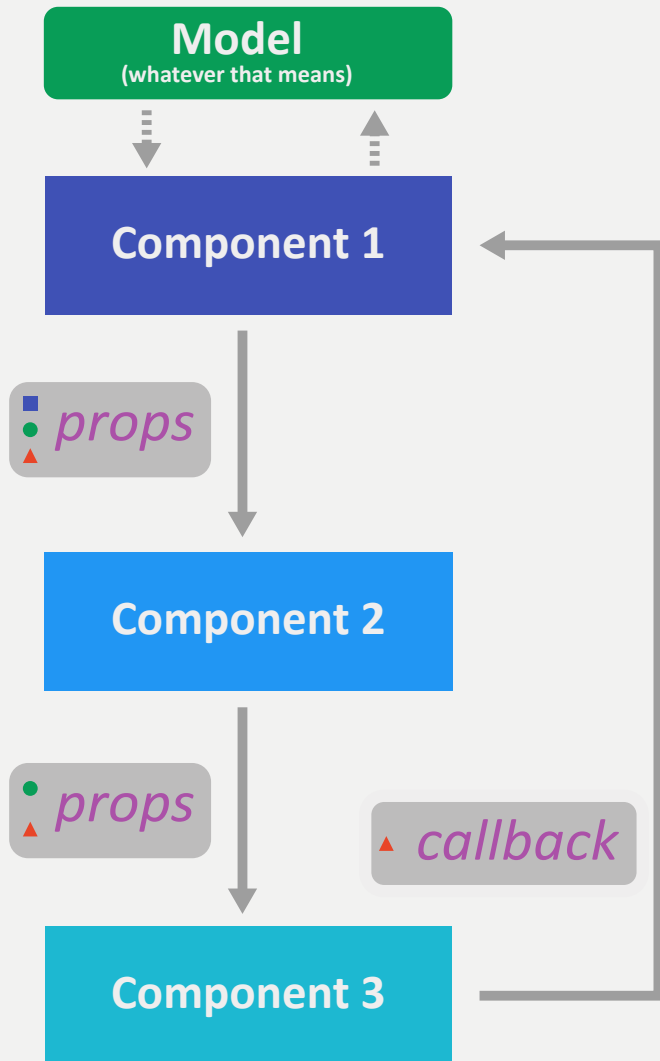
Nils Hartmann (nils@nilshartmann)

REACT MEETUP HAMBURG

WHY FLUX?

Some background first...

SINGLE COMPONENT HIERARCHIE



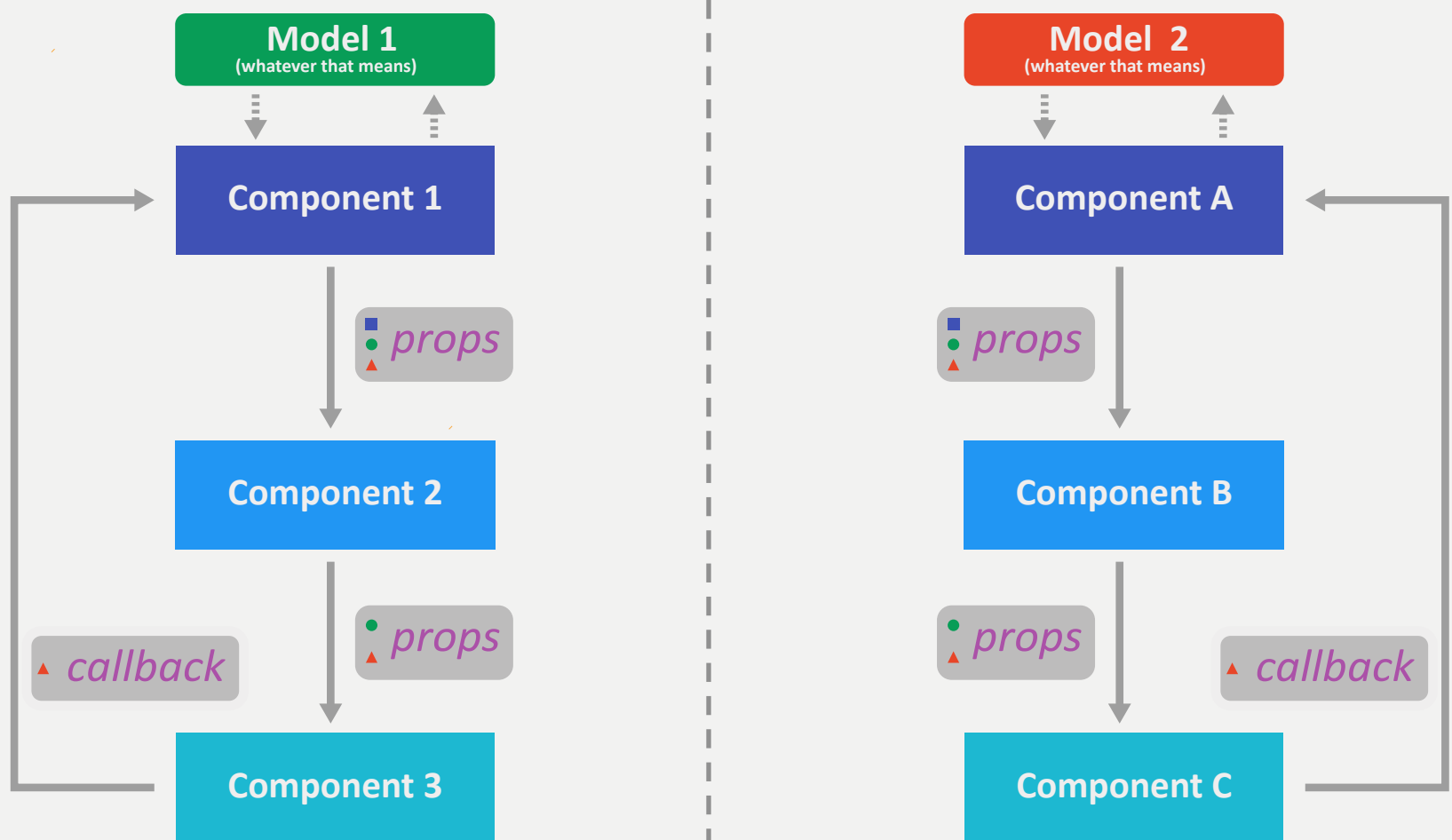
Communication

- Down via properties
- Up via callbacks

Top-level component

- Works on **single** model
- (whatever a ,model' is)

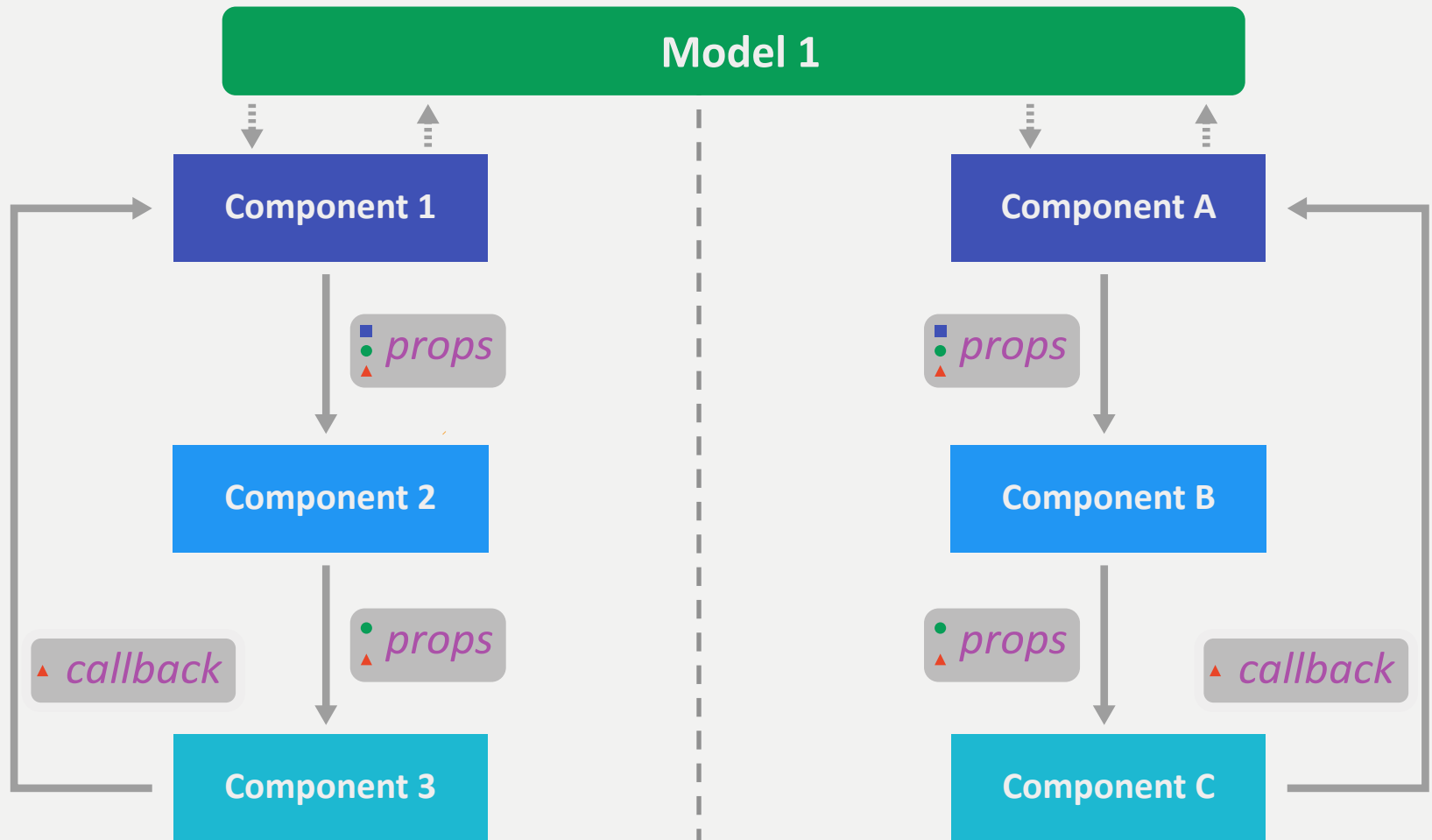
MULTIPLE COMPONENT HIERARCHIES (I)



Isolated Models with independent component trees

- Each component (tree) works on its own model
- Peaceful co-existence

MULTIPLE COMPONENT HIERARCHIES (II)



Shared Model with *independent* component trees

- Each component (tree) works on the **same** model
- Can be confusing: who's taking responsibility? Can lead to back loops

REACT MEETUP HAMBURG

FLUX

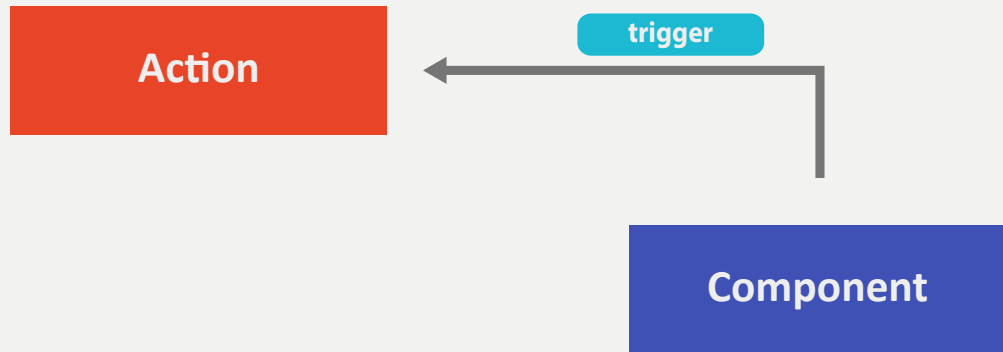
FLUX

„Application architecture for building user interfaces“

- <http://facebook.github.io/flux/>
- Design pattern, not a framework
- Various implementations and interpretations
- Not tied to React

Introduces **unidirectional data flow**

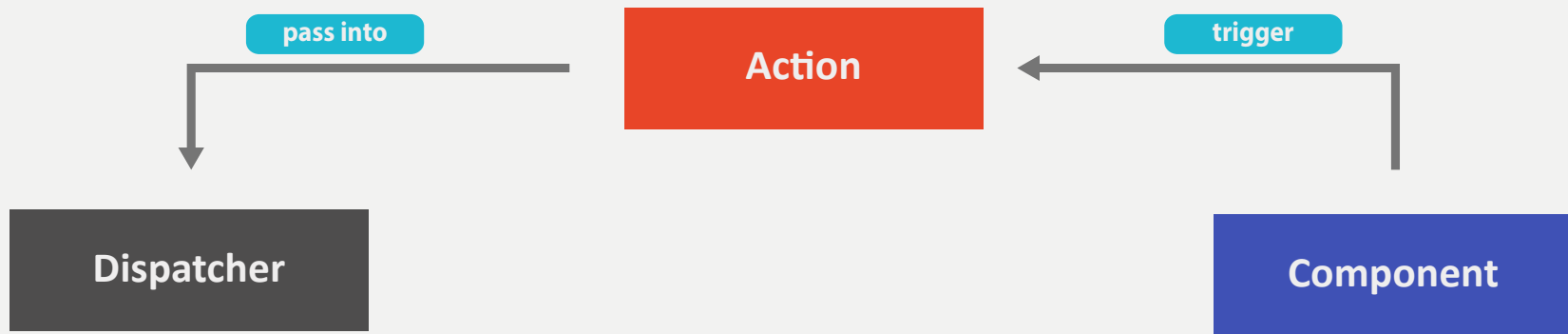
UNI-DIRECTIONAL DATA FLOW - 1



Component triggers an **Action** using an **ActionCreator**

```
<button onClick={() => {  
  UserActionCreators.addUser(...);  
}} />
```

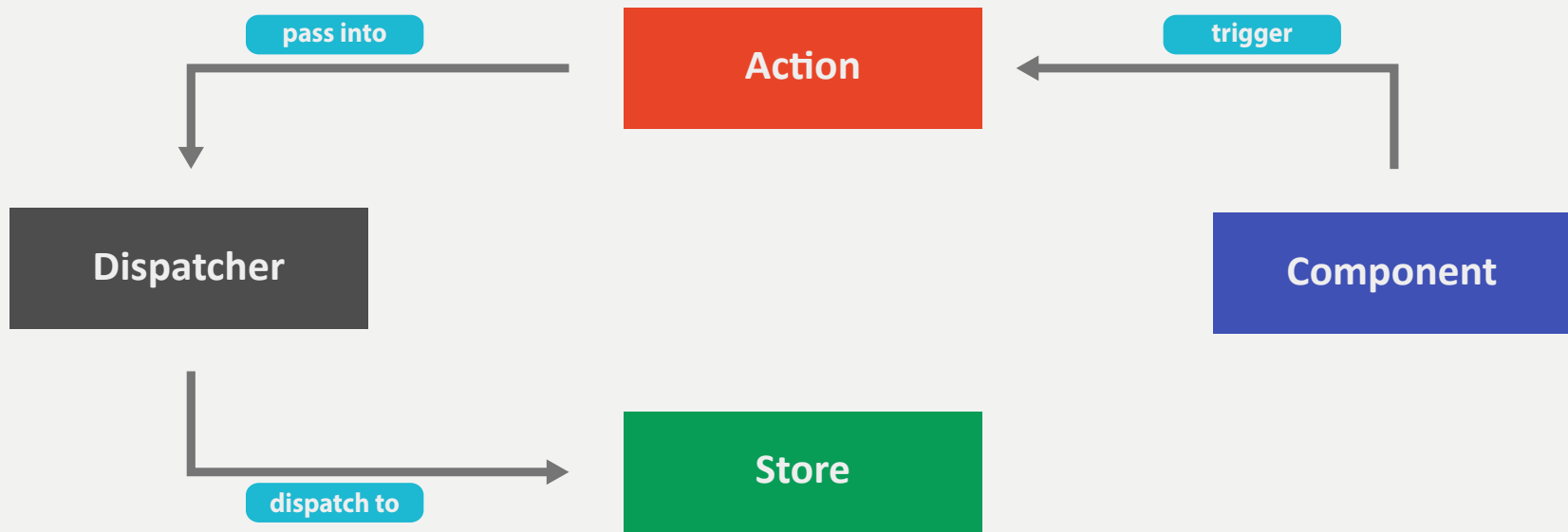

UNI-DIRECTIONAL DATA FLOW - 2



ActionCreator sends **Action** to Central Dispatcher

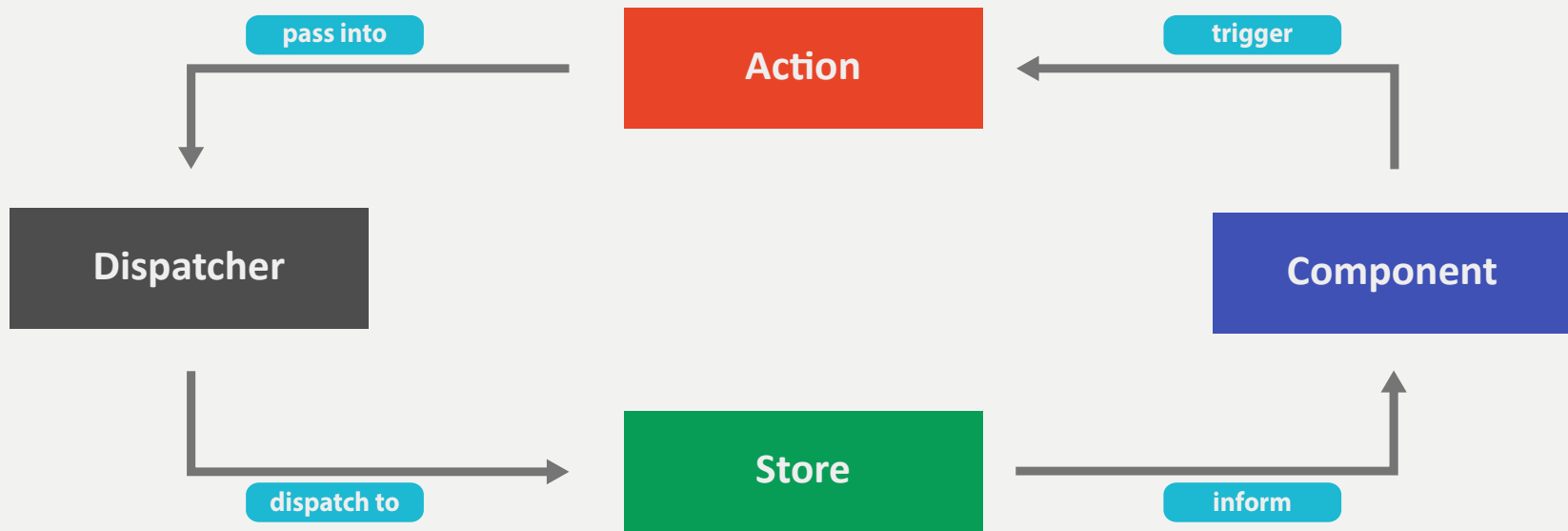
```
addUser(name) {  
  Dispatcher.dispatch({  
    type: 'USER_ADD_ACTION',  
    payload: { user: name }  
  });  
}  
  
                                (UserActionsCreator)
```

UNI-DIRECTIONAL DATA FLOW - 3



Dispatcher forwards the **Action** to **Store**

UNI-DIRECTIONAL DATA FLOW - 4

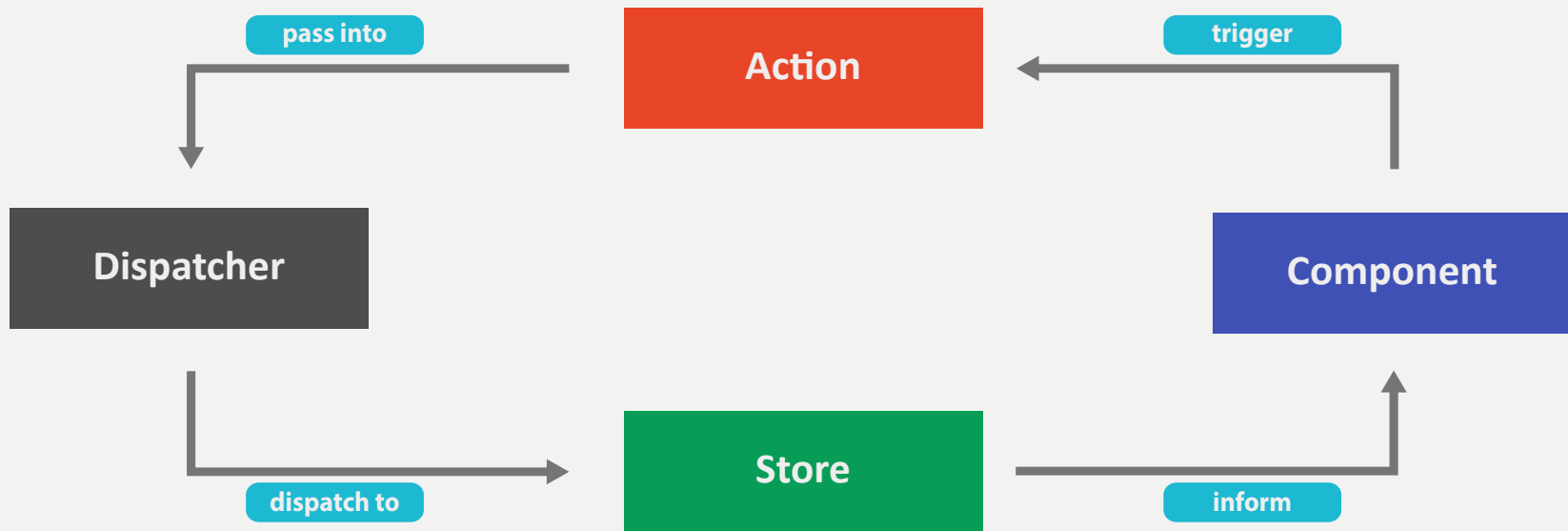


Store processes **Action** and emits a change event

```
handleAction(action) {  
  if (action.type === 'USER_ADD_ACTION') {  
    this.users.push(action.payload.user);  
    emitStoreChangeEvent();  
  }  
}
```

(UserStore)

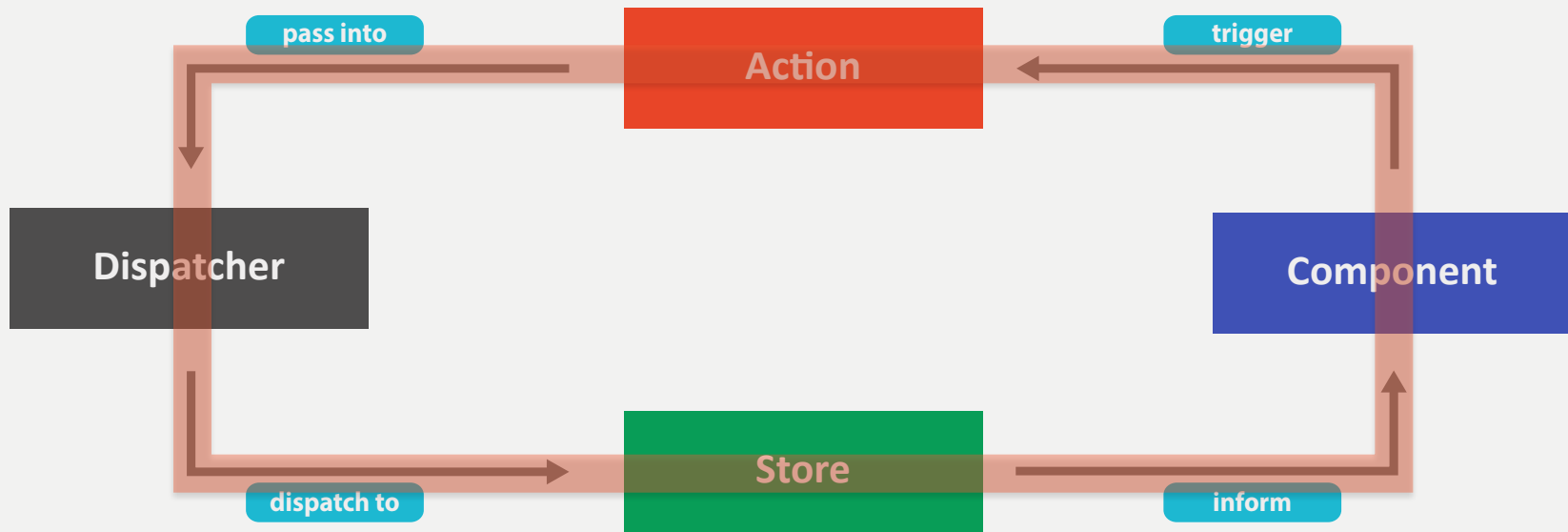
UNI-DIRECTIONAL DATA FLOW - 5



Component receives event and updates its state

```
onUserStoreChange() {  
  this.setState(  
    {users: UserStore.getAllUsers();}  
  );  
}
```

UNI-DIRECTIONAL DATA FLOW - 5



Uni-directional data flow!

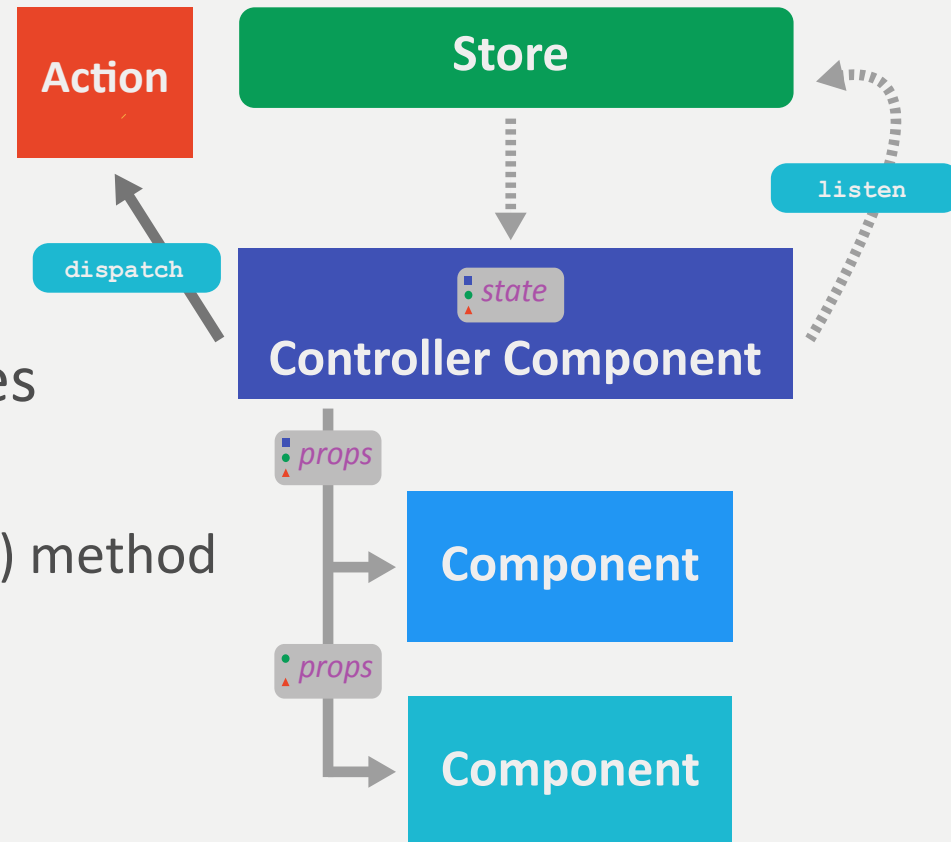
FLUX ELEMENTS IN DETAILS

Note: different implementations interpret differently

COMPONENT

„Regular“ React component

- Triggers an action
 - e.g. on user interaction
- Listens to one or more Stores
- Derives its state from Store
 - Update children from render() method



Controller Component

- Top-level component should be the only component interacting with Flux

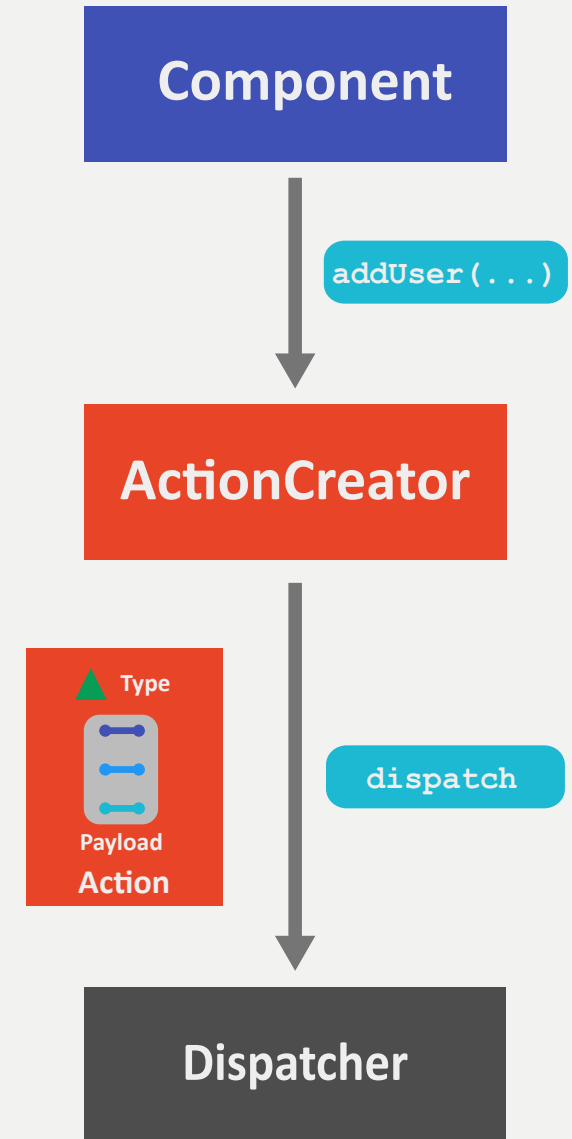
ACTION and ACTIONCREATOR

Action

- Represents *semantic* event happend in the app
- Has an identifying „type“
- Contains information what happend (payload)
- Triggered by Components

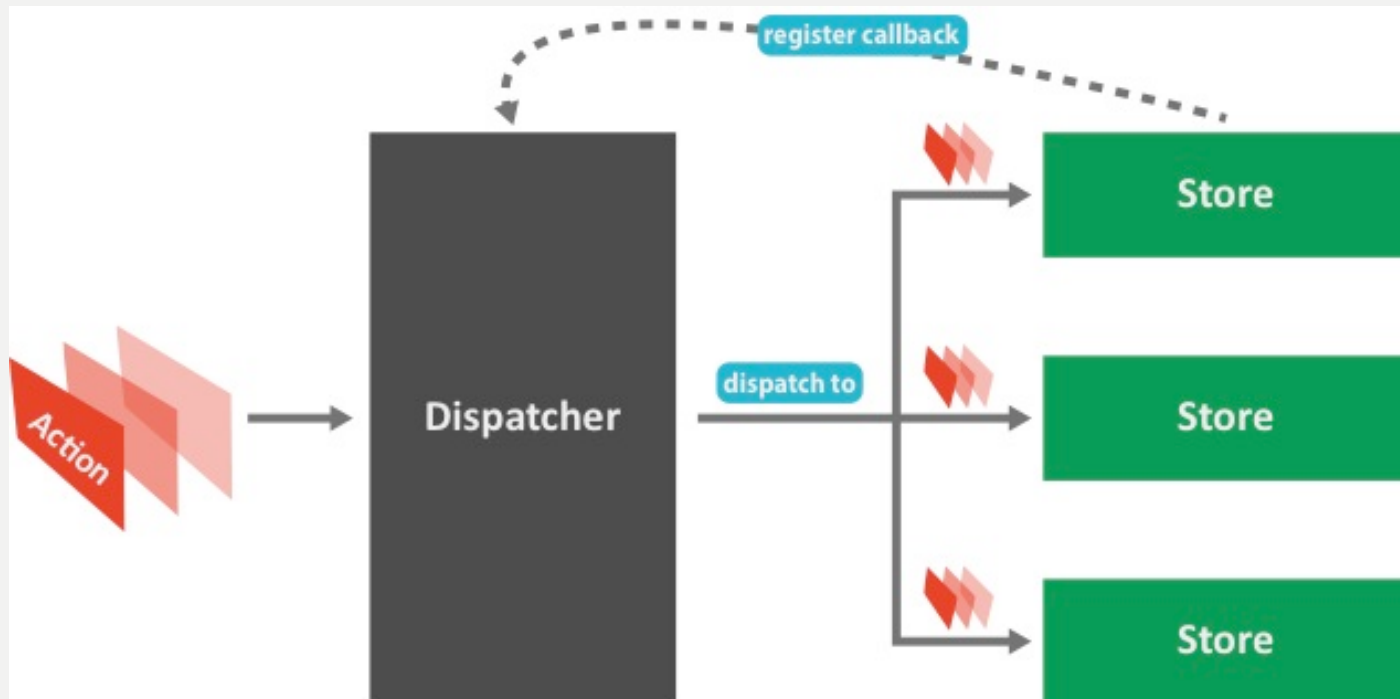
ActionCreator

- Factory for Actions
- Ensures integrity of action object,
- Semantic methods
- Pass Action to Dispatcher



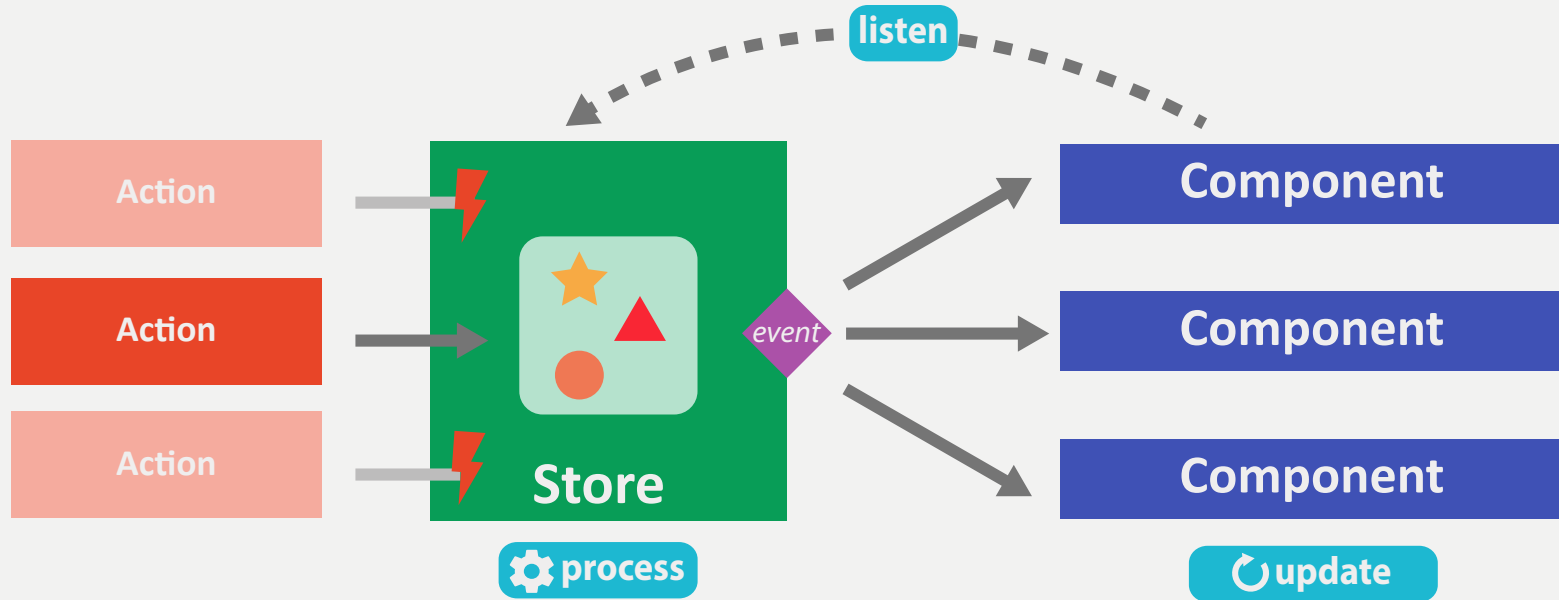
DISPATCHER

- *Pure technical* component, no business logic
- Only one Dispatcher in your application (singleton)
- Forwards Actions to *all* registered Stores
- Works synchronous

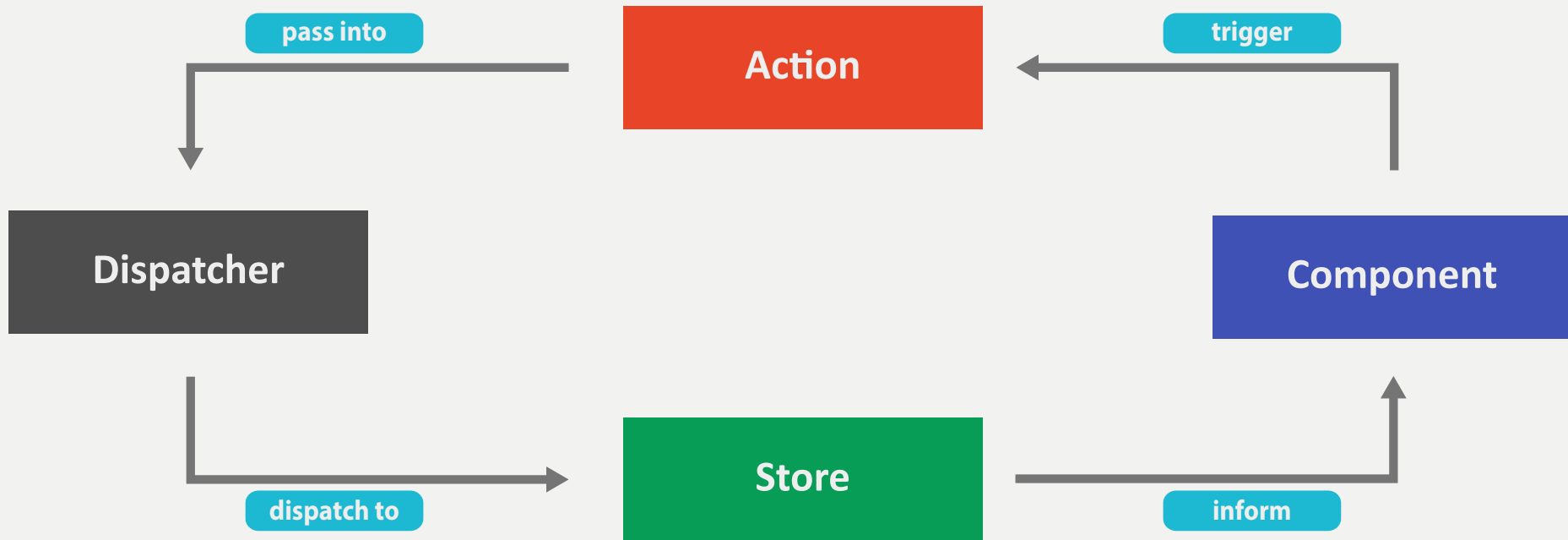


STORE

- Contains the **business logic** for a specific domain
- Connect to the central Dispatcher via callback method
- Process only Actions they are interested in (eg filter by type)
- Emit events after changing model

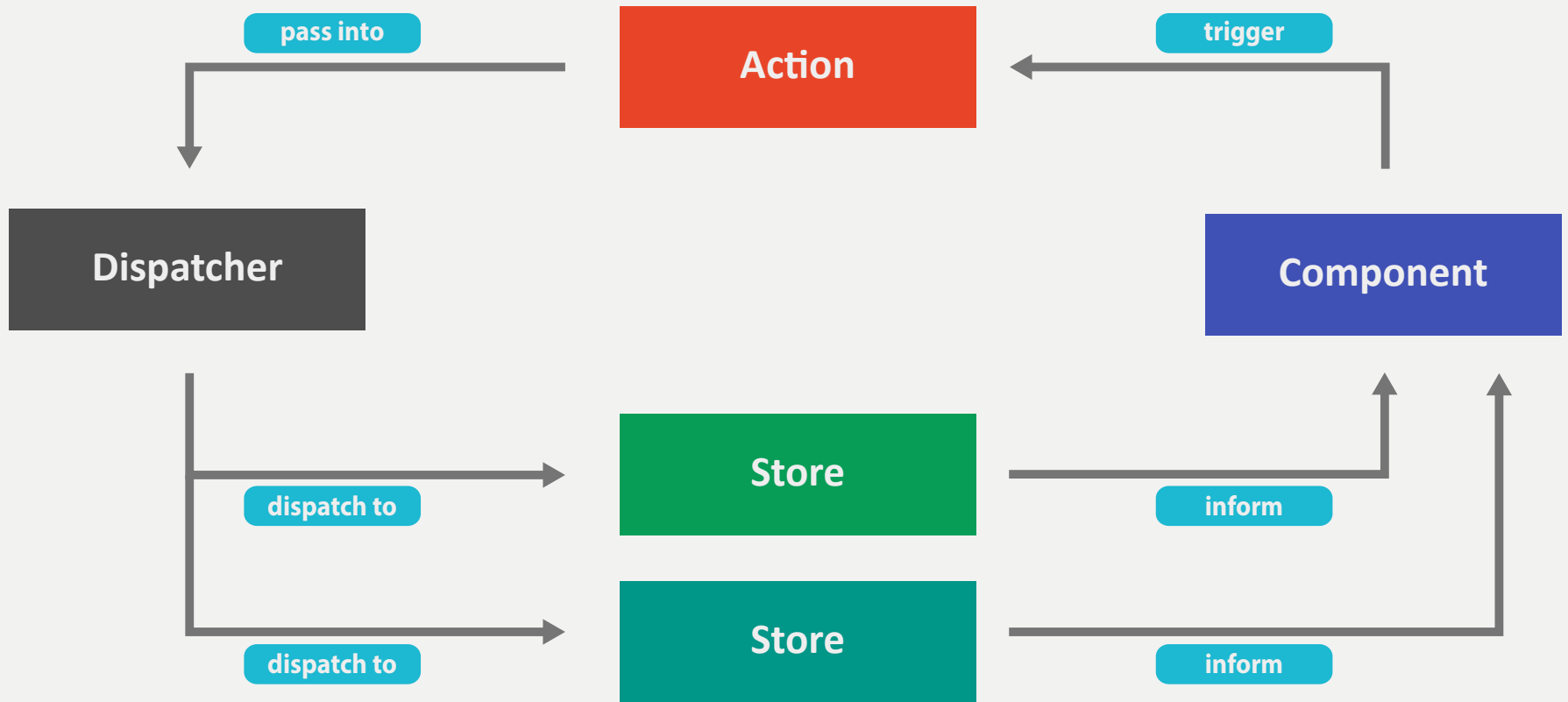


UNI-DIRECTIONAL DATA FLOW - EXAMPLES



Very flexible scenarios...

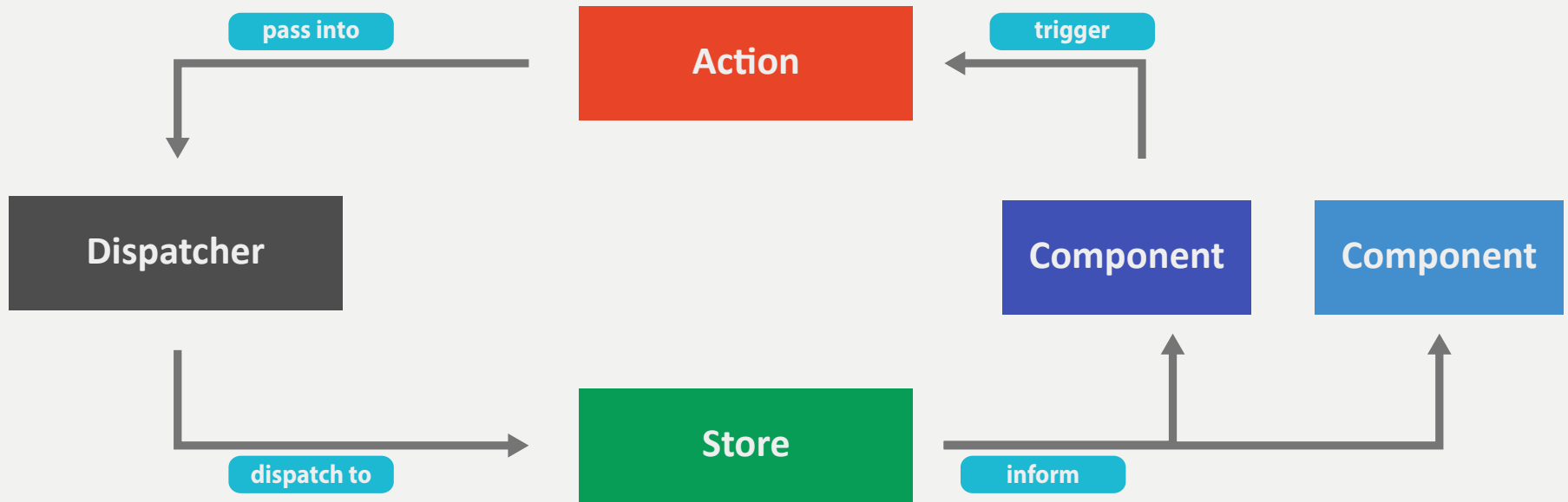
UNI-DIRECTIONAL DATA FLOW - EXAMPLE 1



...Action can be dispatched to multiple stores

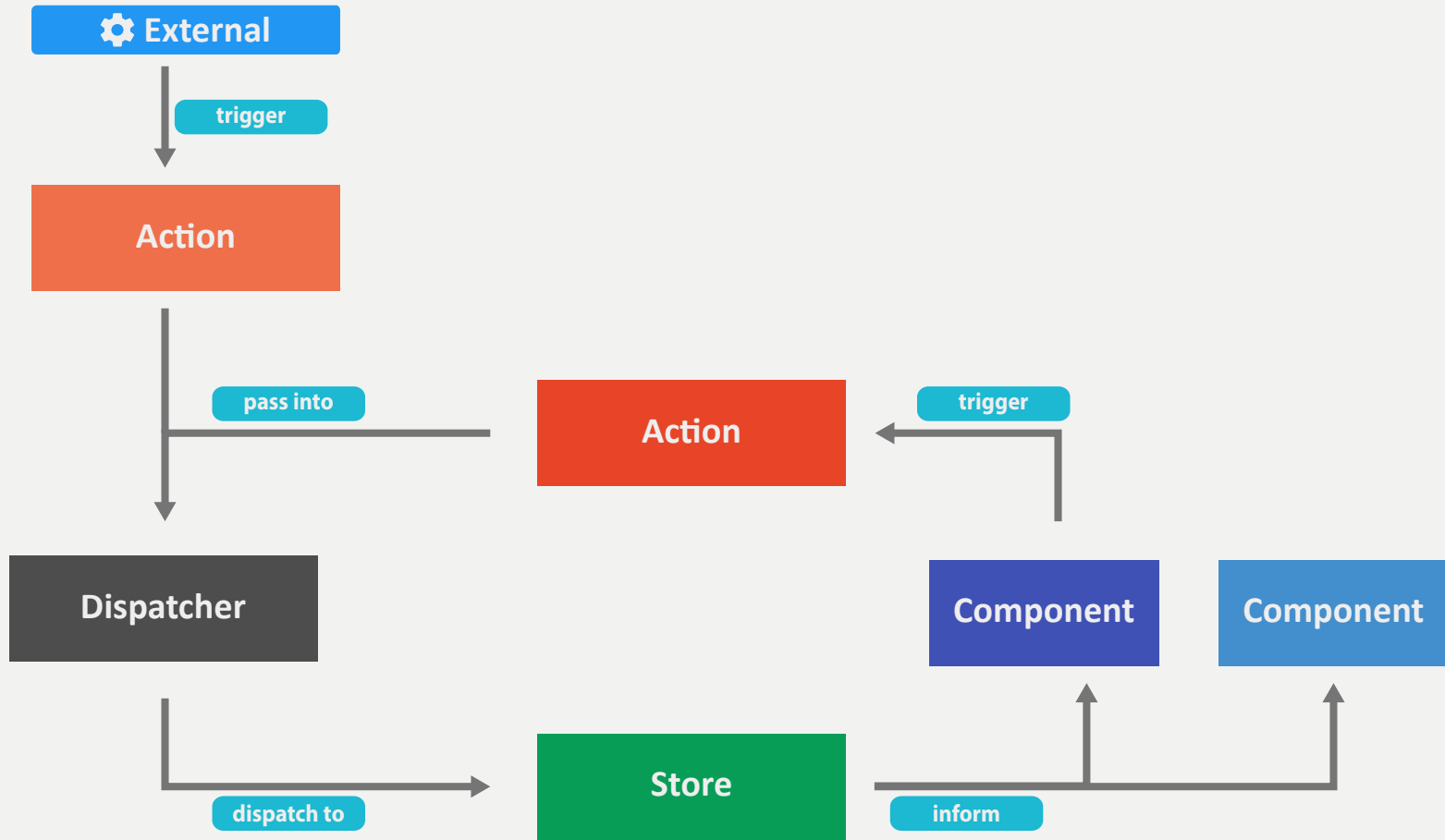
...A component can listen to multiple stores...

UNI-DIRECTIONAL DATA FLOW - EXAMPLE 2



...Multiple components can listen to same store, working on same domain „model“...

UNI-DIRECTIONAL DATA FLOW - EXAMPLE 3



...“external events” (e.g. response from server) can trigger actions

- Flow is still the same

SUMMARY

Useful when working on same model

Defined flow of data

- Easy to track
- Recordable

Lots of new vocabular

- eg: what is the shape of a Store?

Many implemenations

- No „standard“ implementation

REACT MEETUP HAMBURG

THANK YOU!

Slides:

<http://nilshartmann.net/posts/flux-einfuehrung-react-meetup-hh>
