# Redux

An Introduction

@manoj_nama

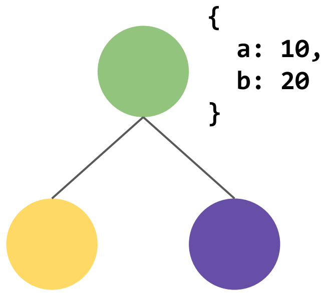
## Current Scenario

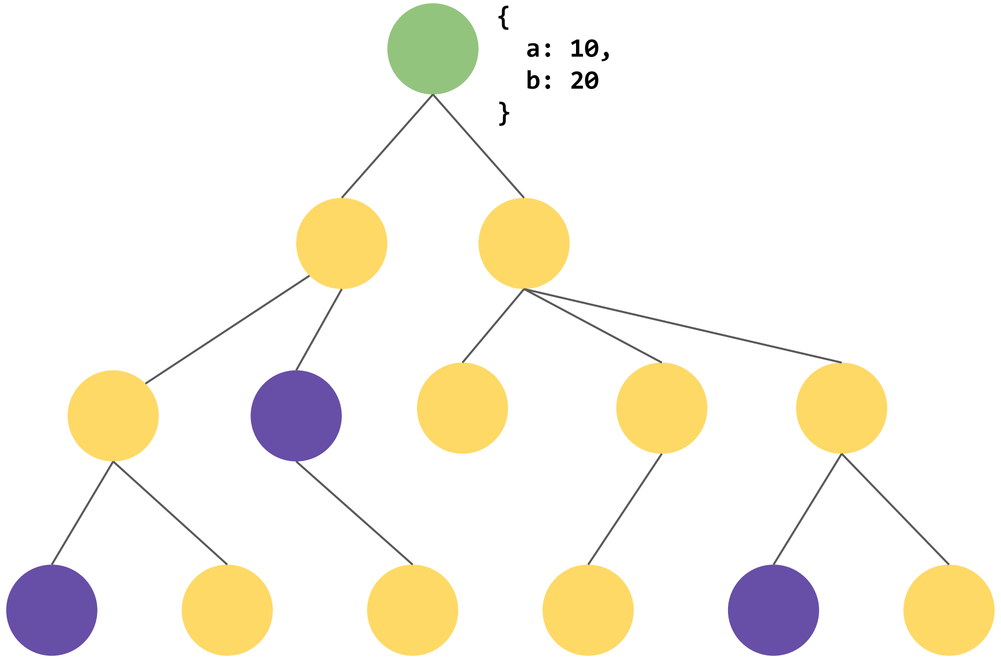
- Storing Data?
- Change Listeners?
- Separate parts of an App? How to decide where data should reside?
- Debugging?

## What is the Problem?

How do you **pass** *data* between your components in a ***Large*** React App?

## What is the Problem?

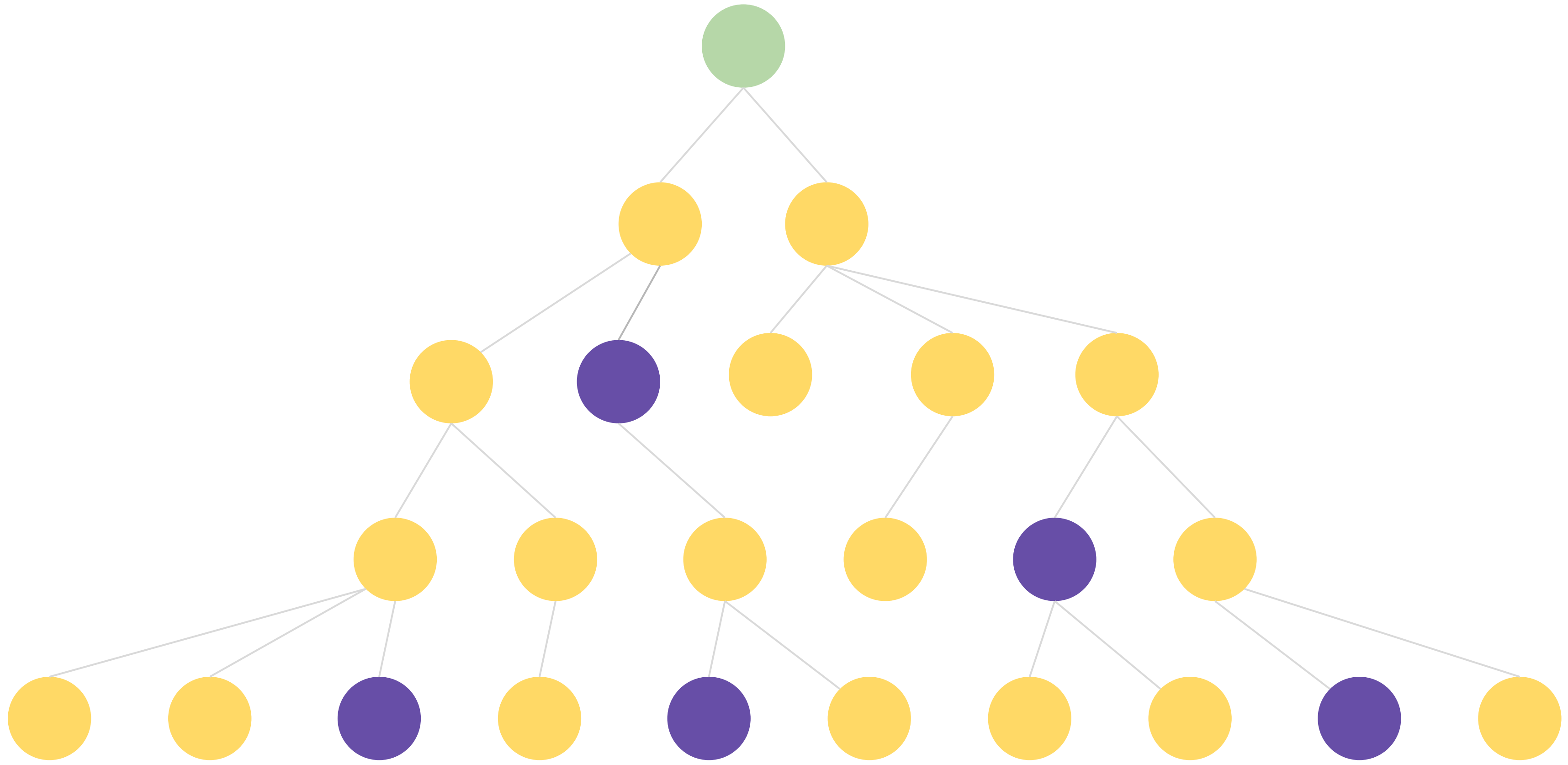




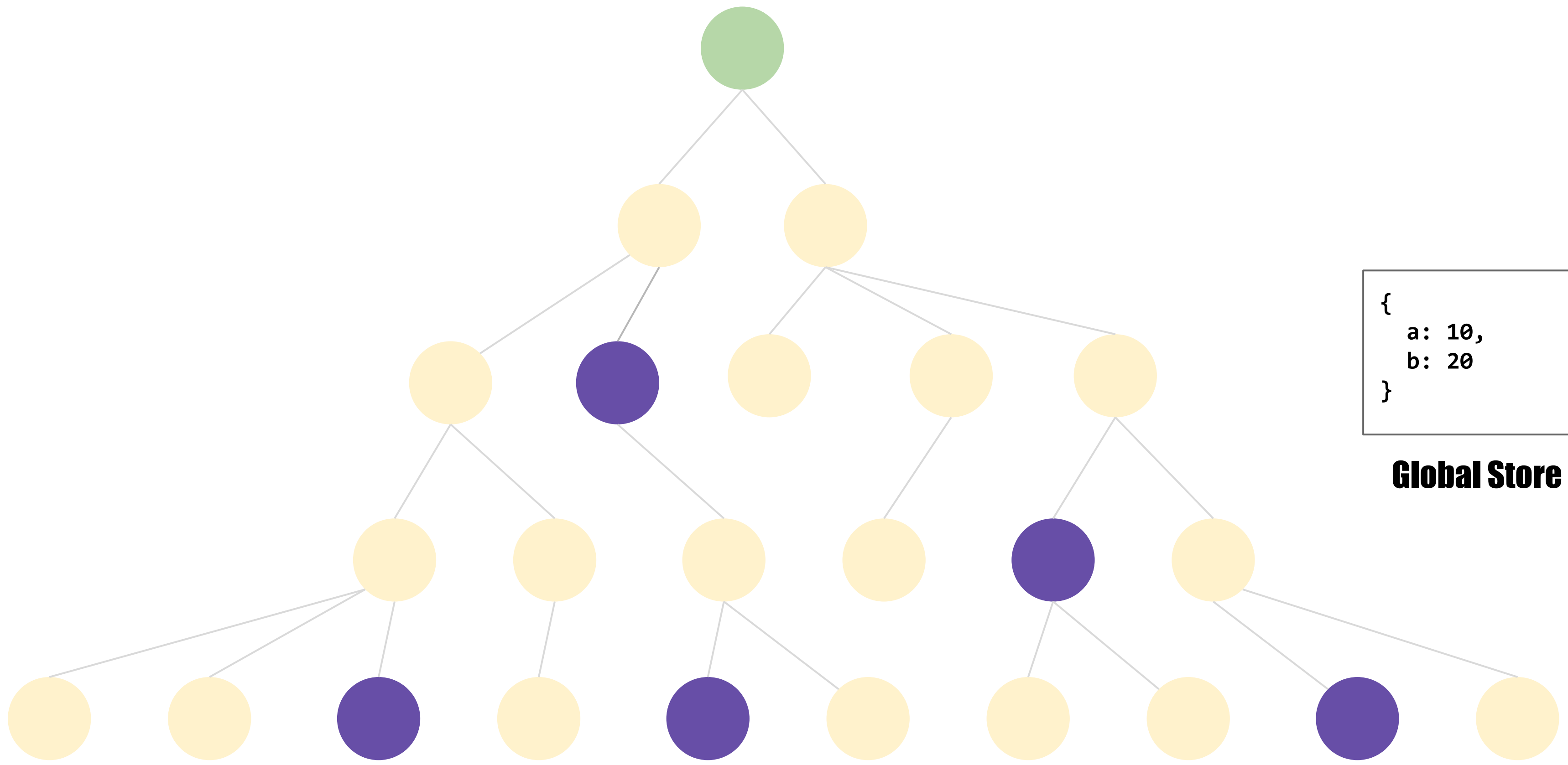
## What is the Problem?

Imagine a **100** *levels for this*?

## How Redux solves this?



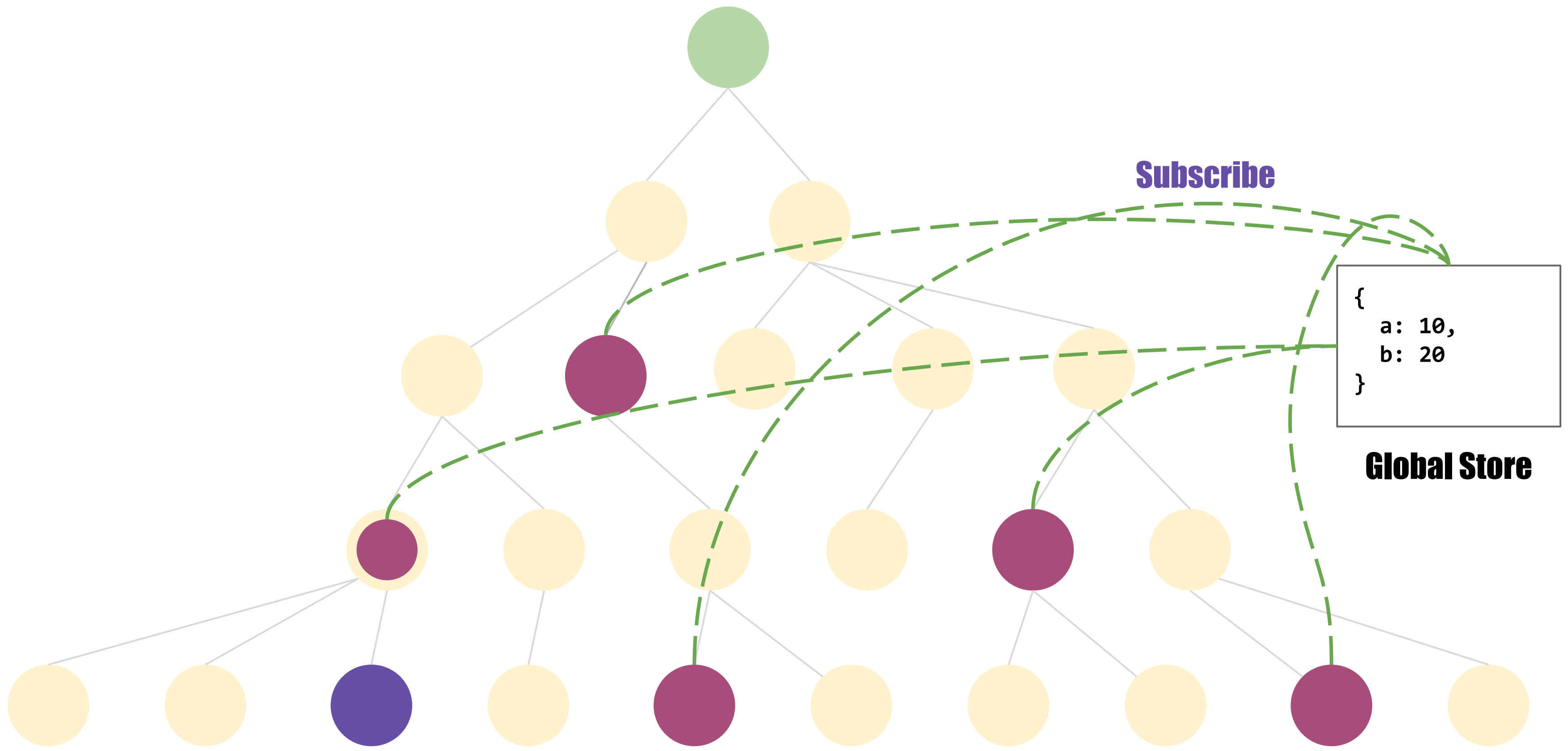
## How Redux solves this?



```
{
  a: 10,
  b: 20
}
```

**Global Store**

# How Redux solves this?



```
{
  a: 10,
  b: 20
}
```

## What is Redux?

It is a **predictable** *state* **container** for JavaScript apps.

– *Official Docs*

## What is Redux?

What the **hell** does that *mean*?

## What is Redux?

It is a **predictable** *state* **container** for JavaScript apps.

– *Official Docs*

## What is Redux?

Separate **Business** and **Presentation** Logic.

*React* for Views, *Redux* for Data

## Understanding Redux

- It is a Glorified Event-Emitter
- It fires *events* when the **store** has changed
- Requires us to keep our data flow **Uni-directional**
- Can be used with Any of the front-end languages, including *Angular*, *Backbone*, *React* and many *more*.
- **Redux** does not have *anything to do* with **React**

## Components of Redux

There are **3** basic/essential ***components*** to keep in mind when using Redux

- Store
- Reducers
- Actions

## Store

A **Global** Object, Holds your **entire** *application* state. To *update* any part of app, **change** the store.

## Store

```
{
    loading: true,
    items: [{...}, {...}],
    user: { email: "...", name: "..." },
    products: [
        { id: 1, ... },
        { id: 2, ... },
        { id: 3, ... },
    ]
}
// normalized state?
```

## Action

A **plain** JavaScript Object, specifies what to do. Fire an **action** when the *store* needs to be *updated*.

## Action

```
{
    type: "FETCH_USERS",  // required
    data: {
        offset: 50,
        limit: 10,
        query: "mike"
    }
}
```

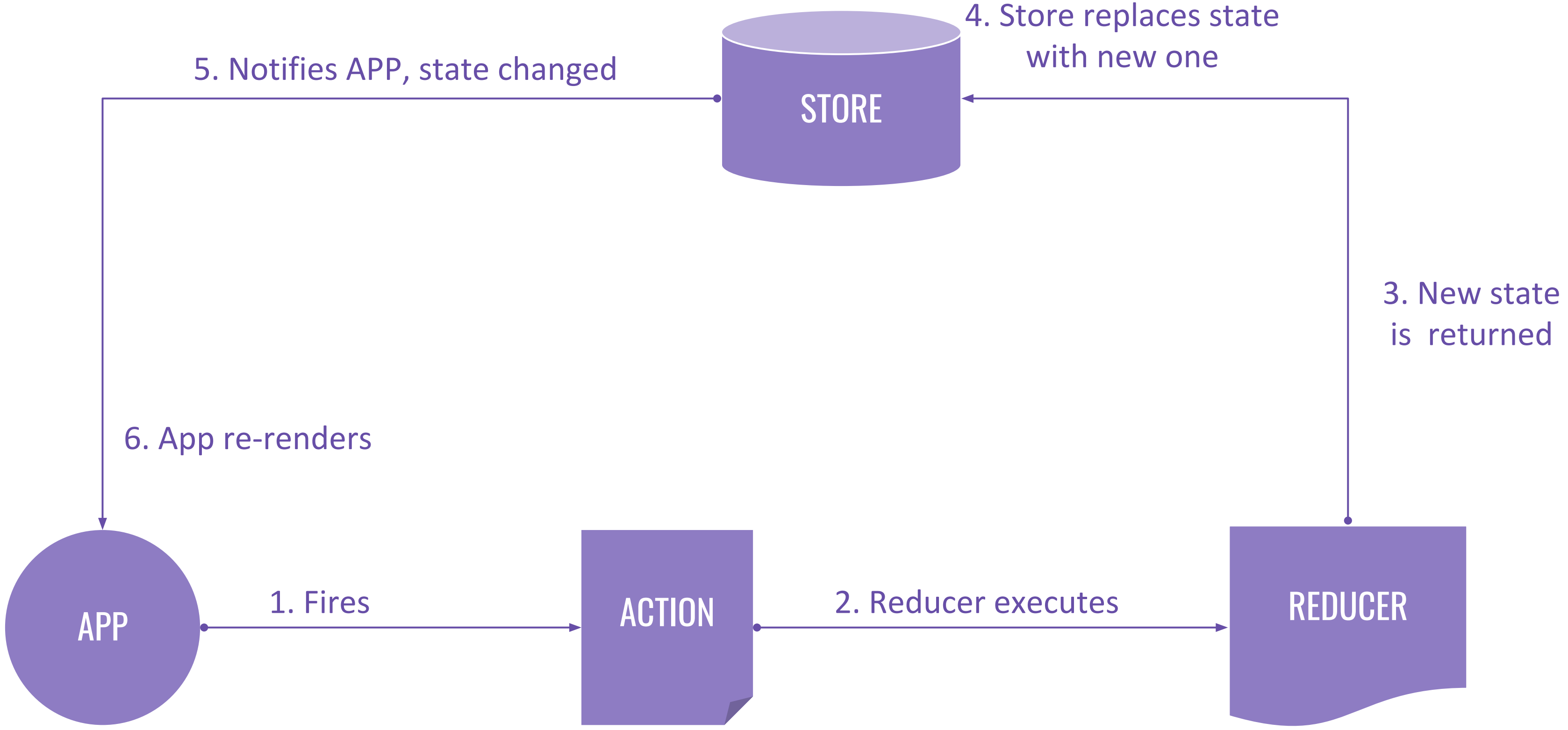
## Reducer

A **pure** function, takes an **action** and **state**, *returns* a **new** state.

## Reducer

```
const reducer = (state, action) => {
    switch(action.type) {
        case "FETCH_USERS_COMPLETE": {
            return {
                users: [...state.users, ...action.users],
                loading: false
            };
        }
        case "...": { ... },
        default: { return state; }
    }
}
```

# Data Flow


4. Store replaces state with new one
5. Notifies APP, state changed
STORE
3. New state is returned
6. App re-renders
APP
1. Fires
ACTION
2. Reducer executes
REDUCER