

Formation REDUX

avec React.js

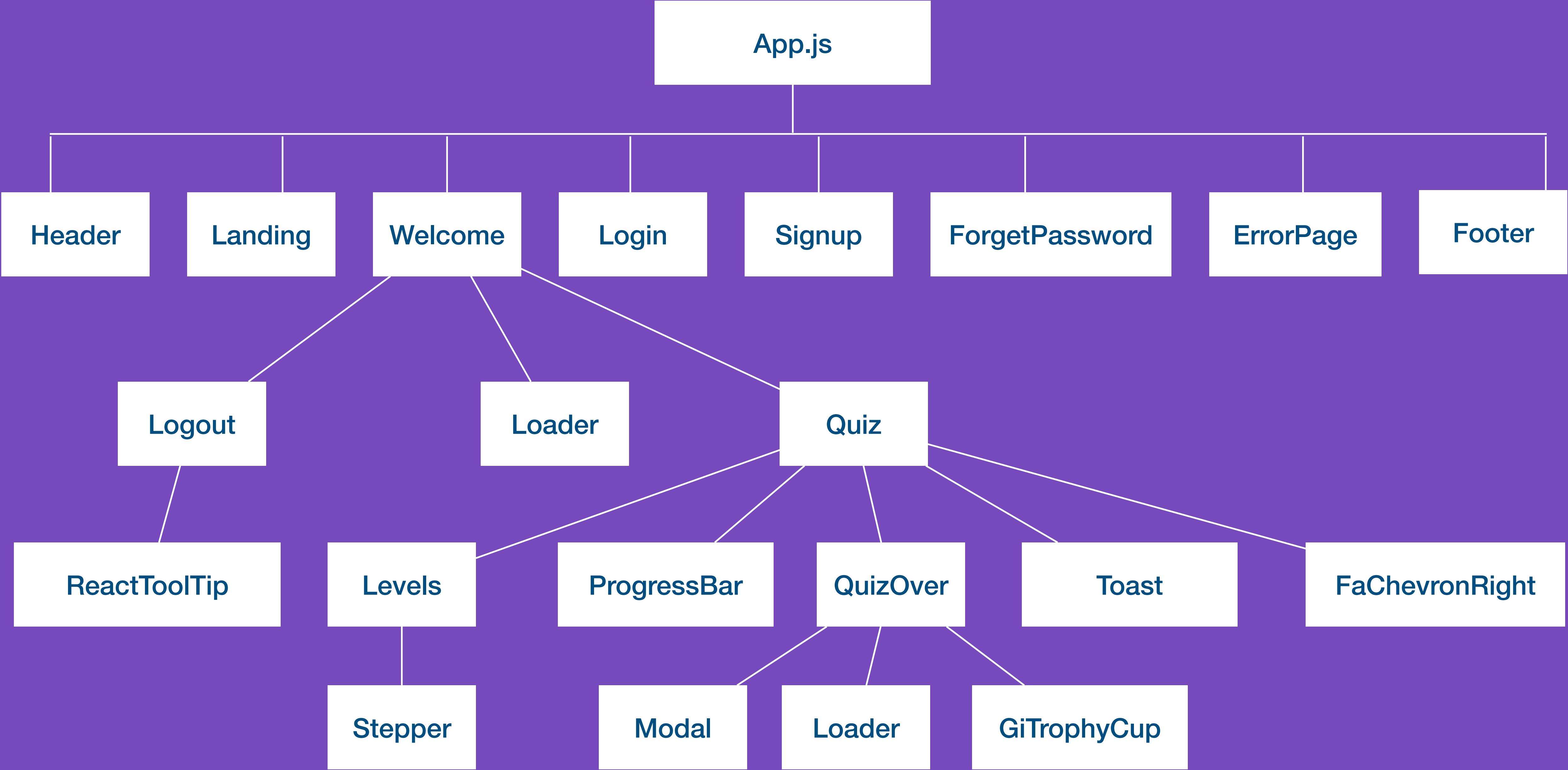


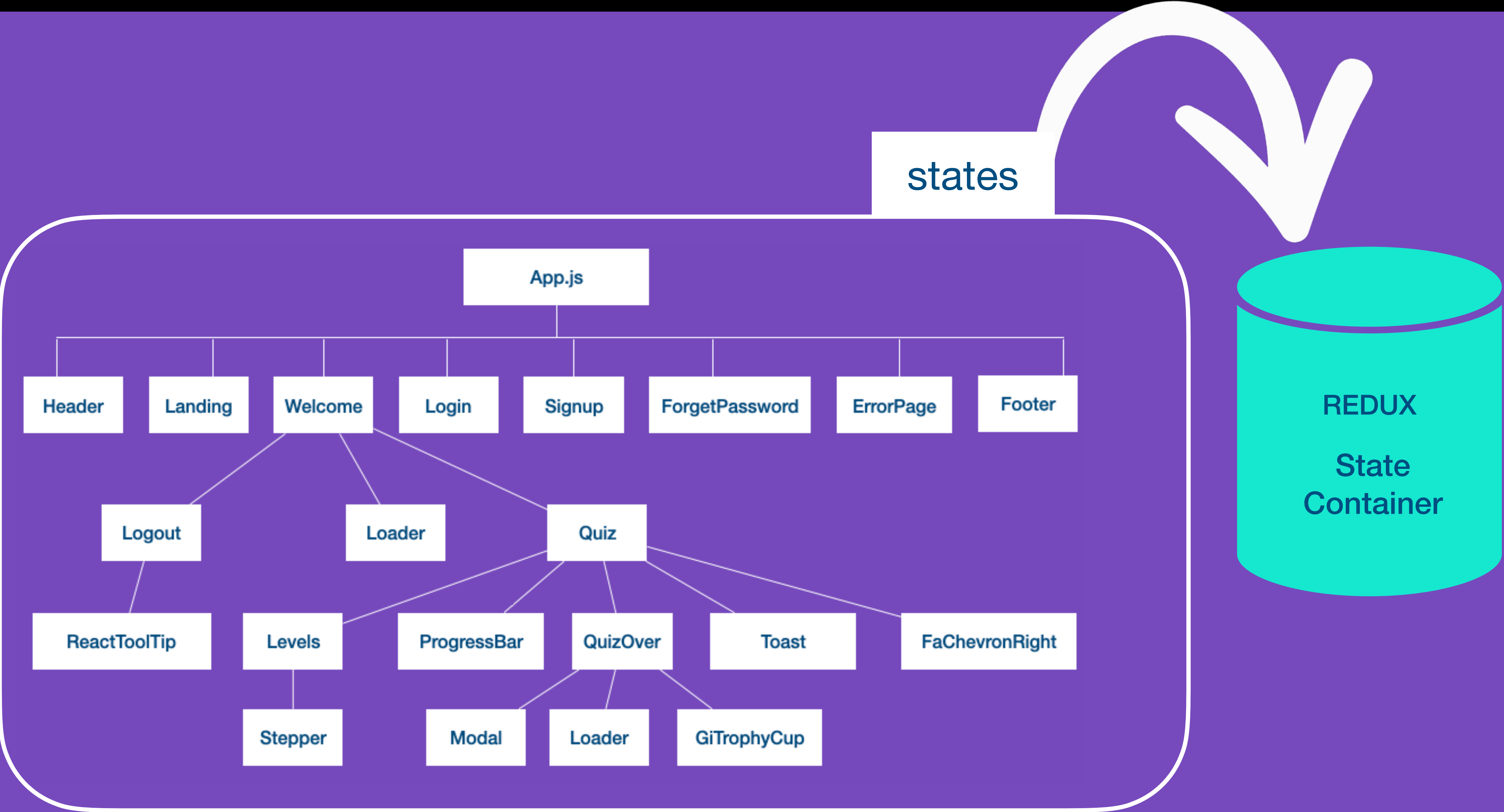
« Redux est un conteneur d'état prévisible pour les applications JS »

Conteneur d'Etat

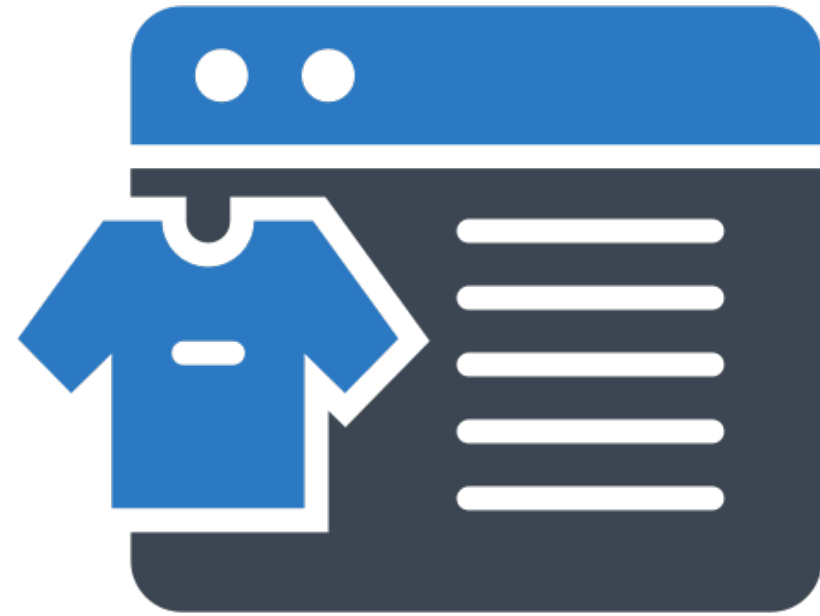
Prévisible

Application JS





STORE



```
state = {  
  numItems: 9  
}
```

ACTION



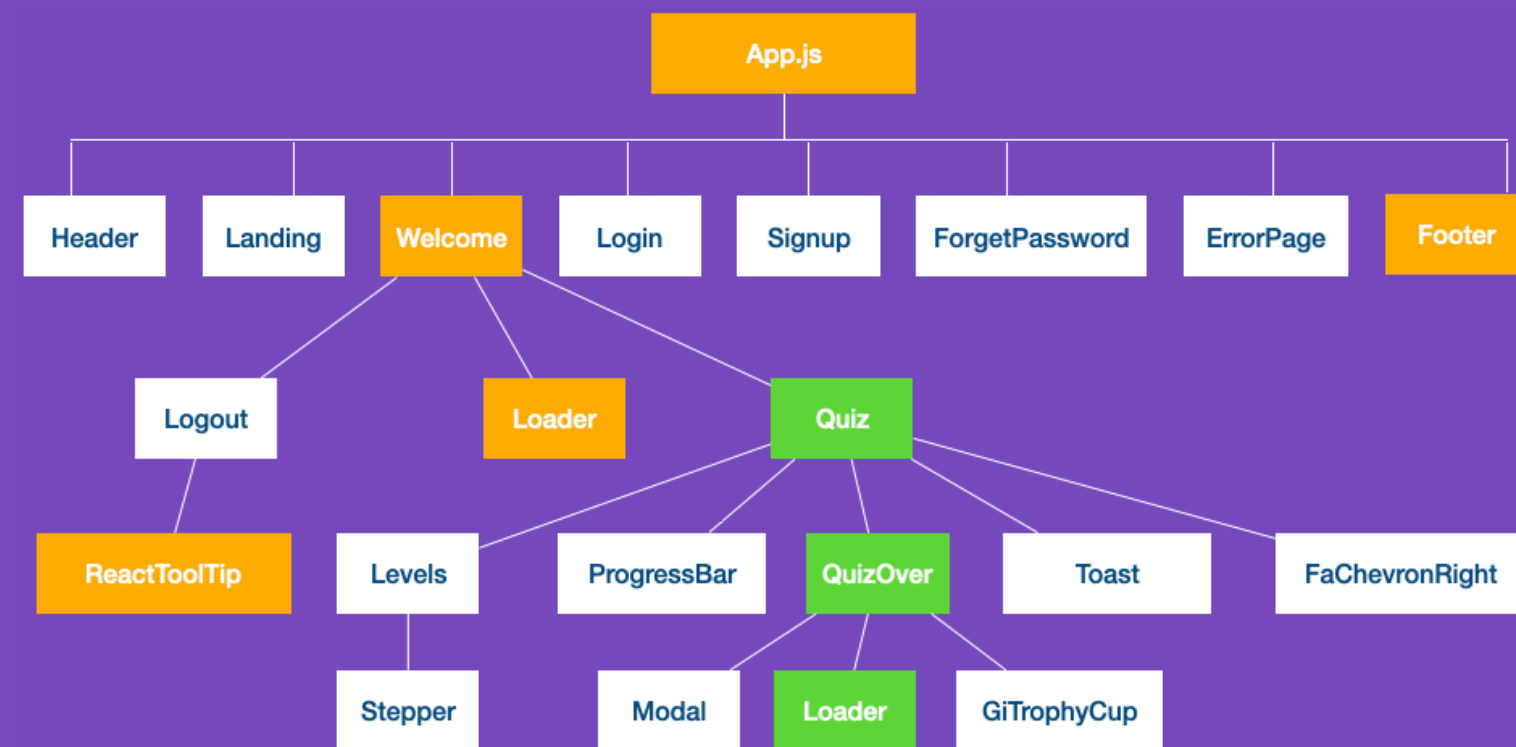
```
{  
  type: 'BUY_ITEM'  
}
```

REDUCER



(prevState, action) => **newState**

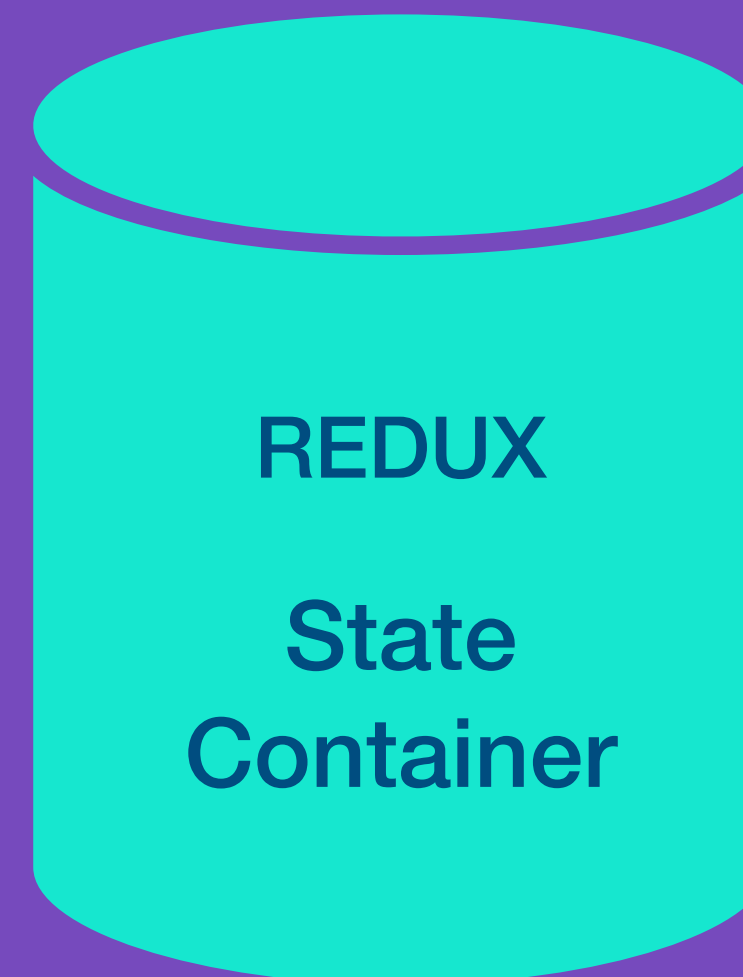
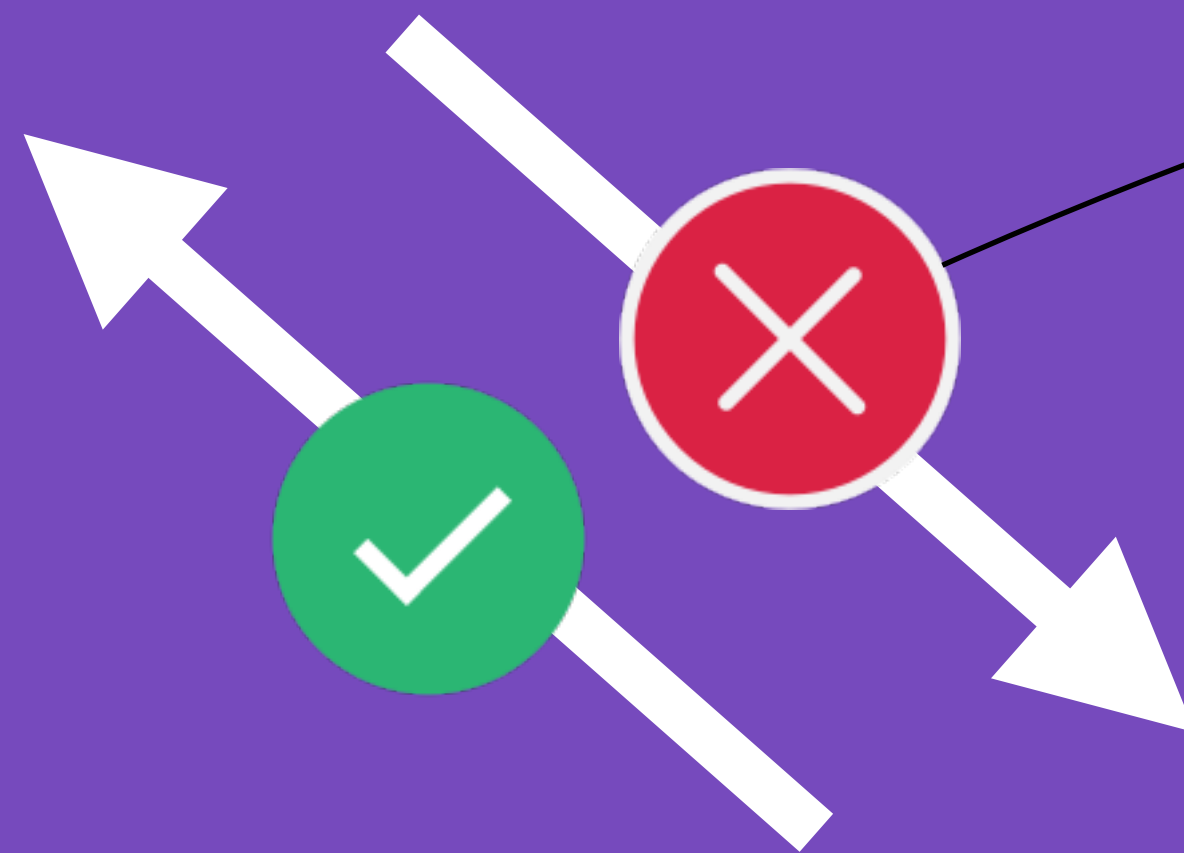
Application - UI

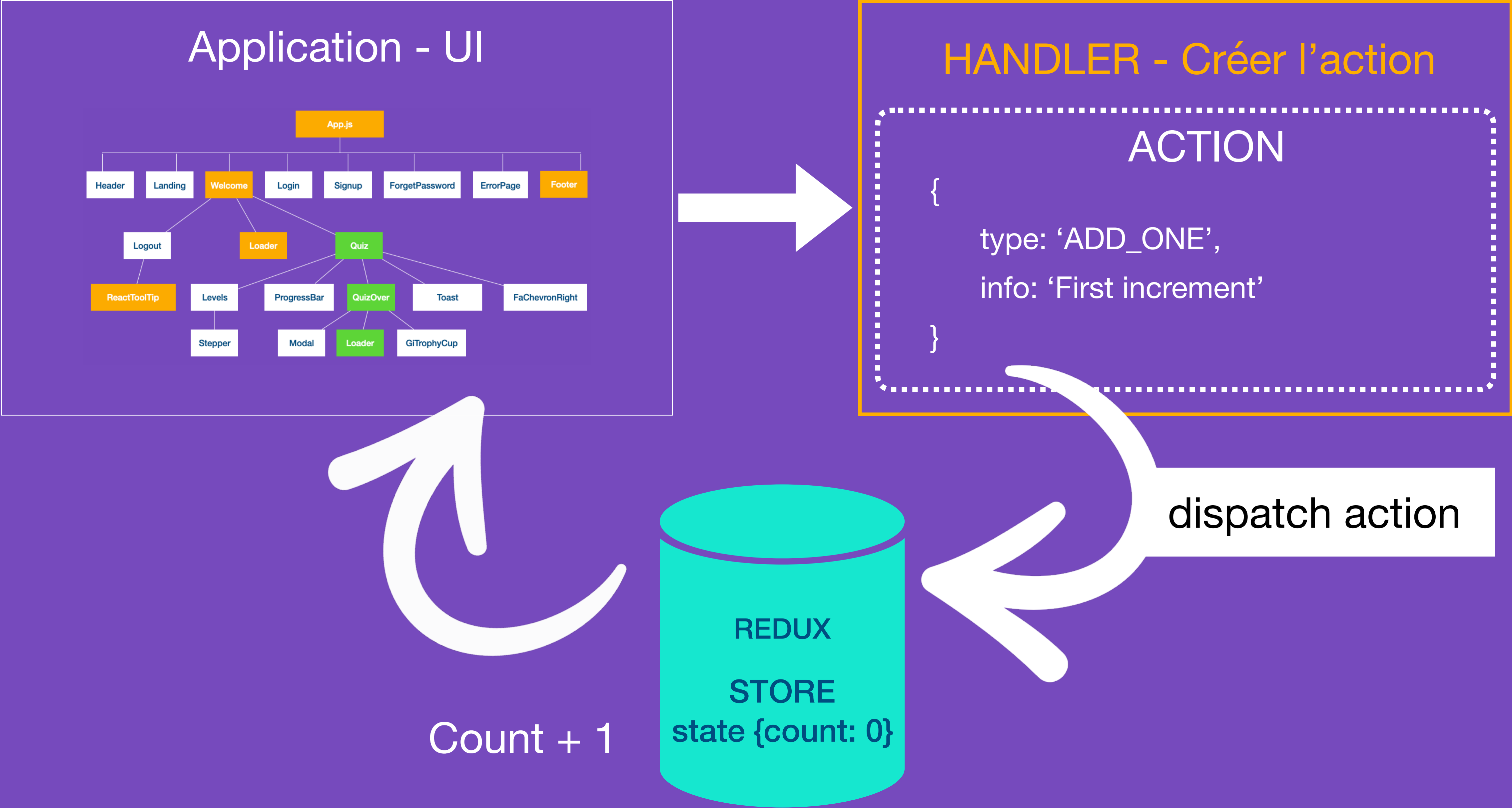


- Mission: Incrémenter « Count » que j'ai dans le state container Redux

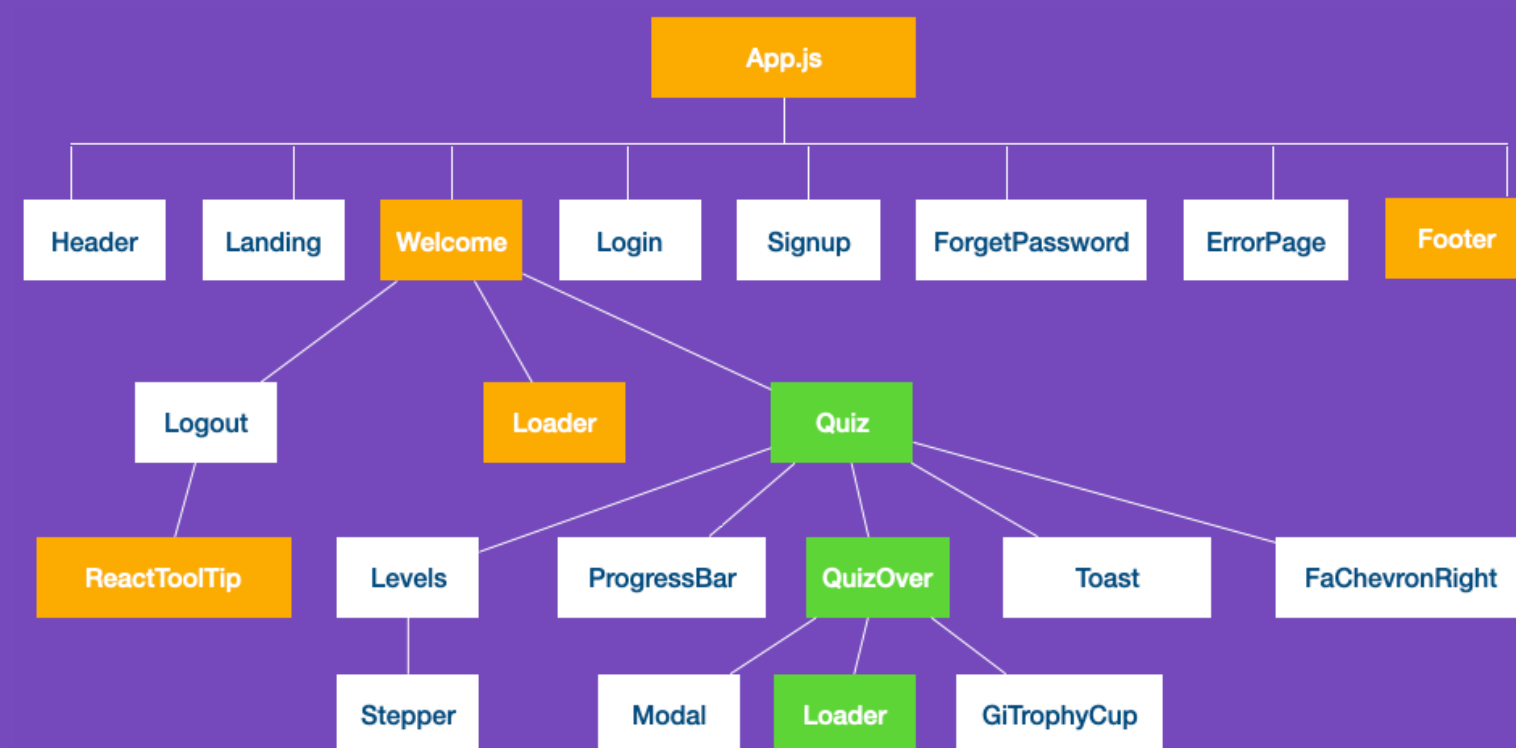


Emettre une action
(Définir une action via un handler)





Application - UI



HANDLER - Créer l'action

ACTION

```
{  
  type: 'ADD_ONE',  
  info: 'First increment'  
}
```

REDUCER (pure function)

(prevState, action) => newState

copie du state

dispatch action

REDUX
STORE
state {count: 0}

Magasin (App)



= 5

Acheter

dispatch une action

HANDLER - Créer l'action

ACTION

{ type: 'BUY_PHONE' }

REDUX STORE

state {
 phones: 5
}

REDUCER

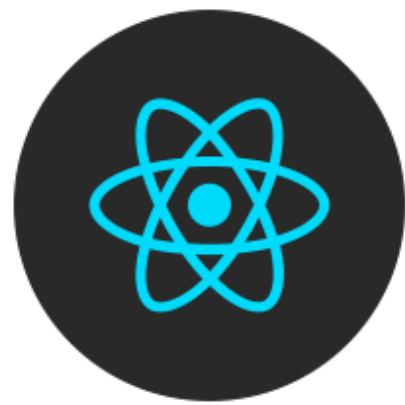
({ phones: 5 }, BUY_PHONE) => newState

BUY_PC
BUY_USB

{ phones: 4 }



Prérequis



Outils



Magasin (App)



= 5

Action

```
{ type: 'string' }
```

REDUCER

(prevState , Action)



```
{ phones: 4 }
```

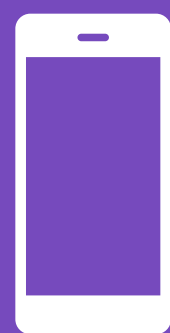


createStore(reducer)

REDUX STORE

```
state {  
  phones: 5  
}
```

Magasin (App)



= 5

REDUX STORE

state { phones: 5 }

getState()

dispatch(action)

subscribe(Listener)

Action
{ type: 'string' }

REDUCER

(prevState , Action)



{ phones: 4 }



Permet l'accès au Store

Effectuer le dispatch d'une action

Le listener se lance à chaque modification du state (dans store)



= 5



= 10



Action
{ type: 'BUY-PHONE' }

Action
{ type: 'BUY-TABLET' }

REDUX STORE

```
state {  
  phones: 5,  
  tablets: 10  
}
```

getState()

dispatch(action)

subscribe(Listener)

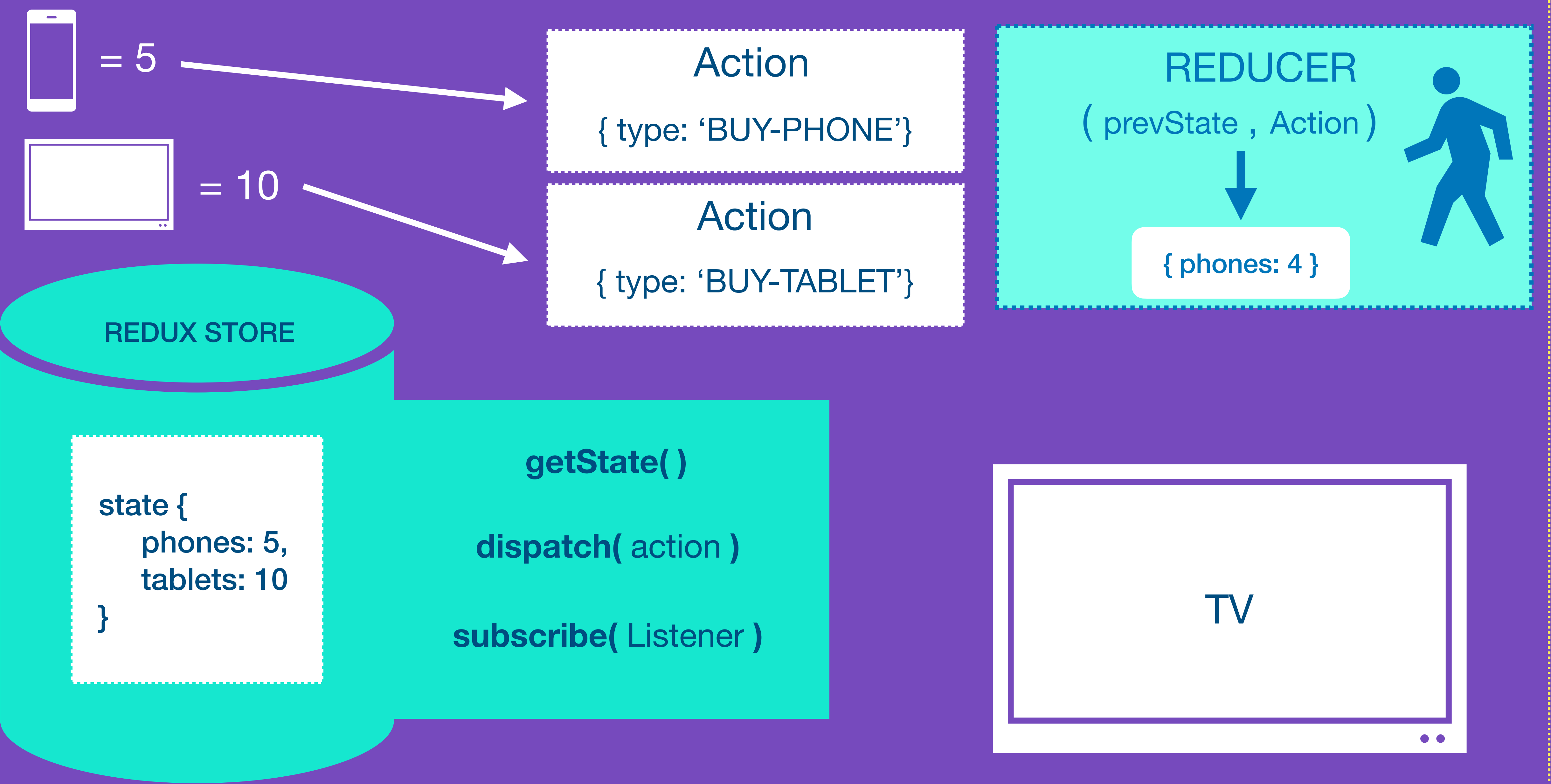
REDUCER

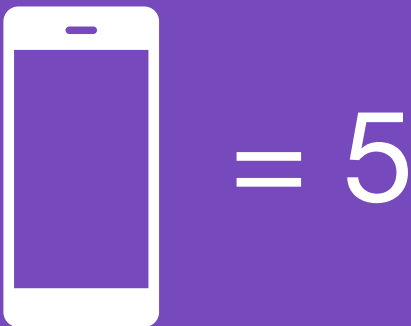
(prevState , Action)



{ phones: 4 }







= 5

Action

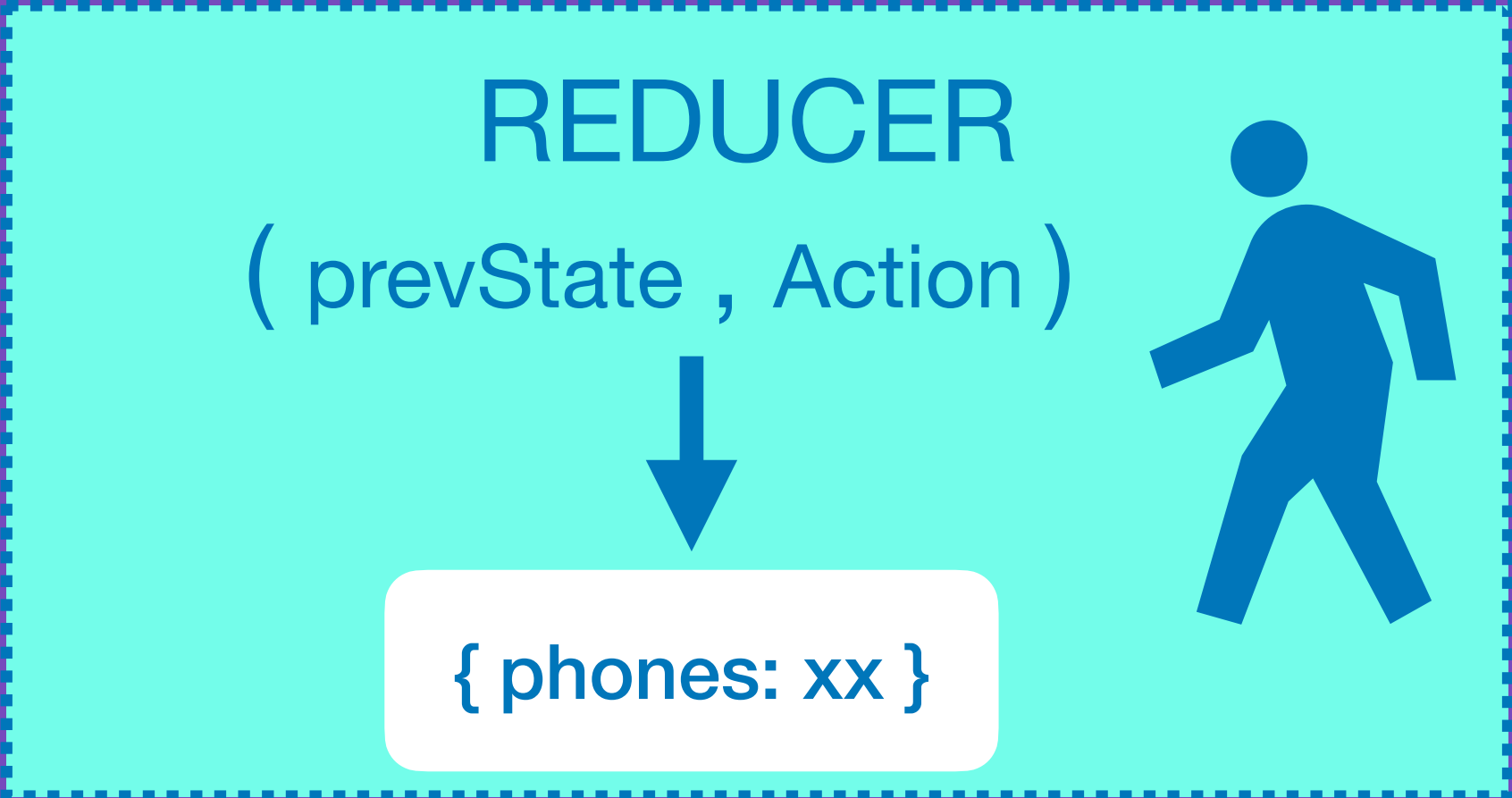
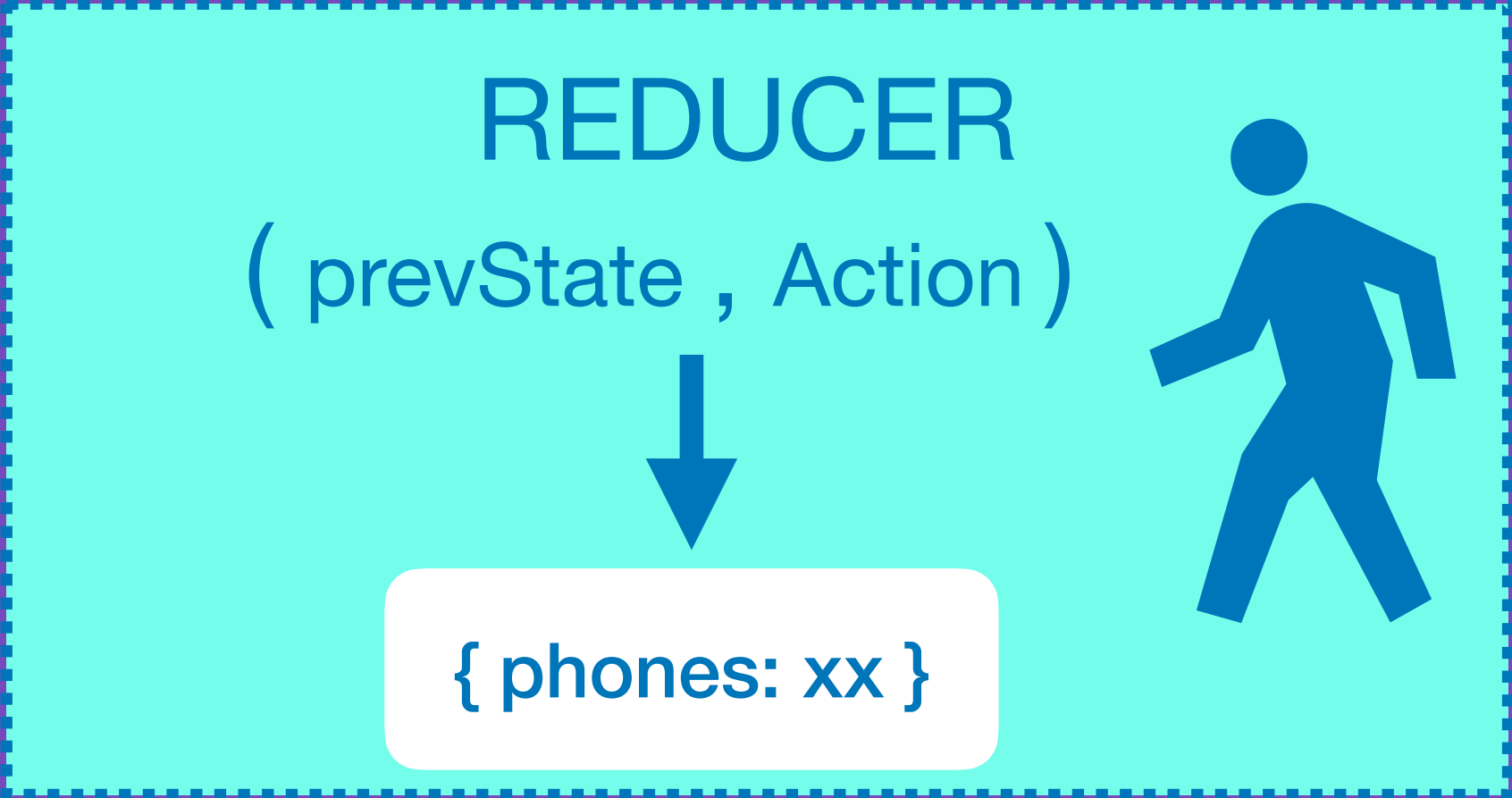
{ type: 'BUY-PHONE' }



= 10

Action

{ type: 'BUY-TABLET' }



= 20

Action

{ type: 'BUY-TV' }



createStore(reducer)

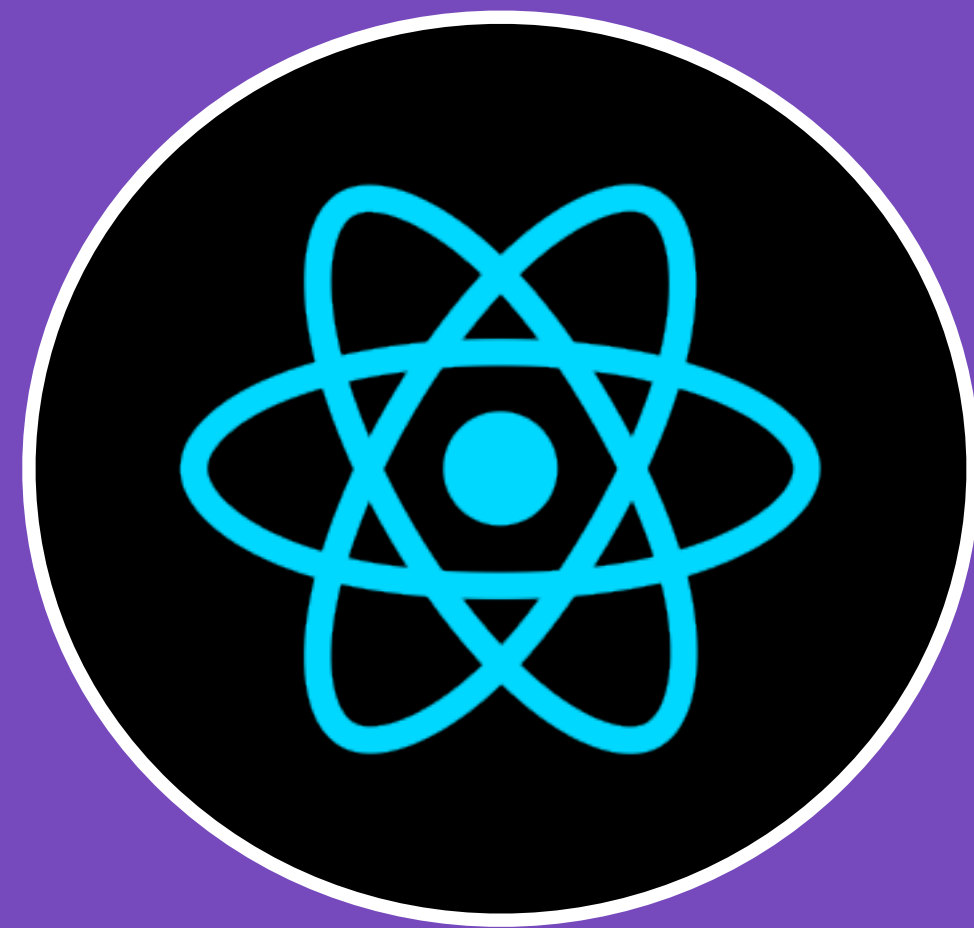


reducer1 reducer2
 ↓ ↓
combineReducers({ key: value, key: value })



rootReducer

REACT



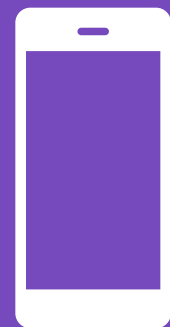
REDUX



REACT REDUX

Attention: Pour continuer, vous devez maîtriser les fondamentaux de React !

Magasin (App)



= 5

REDUX STORE

state { phones: 5 }

getState()

dispatch(action)

subscribe(Listener)

Action
{ type: 'string' }

REDUCER

(prevState , Action)



{ phones: 4 }

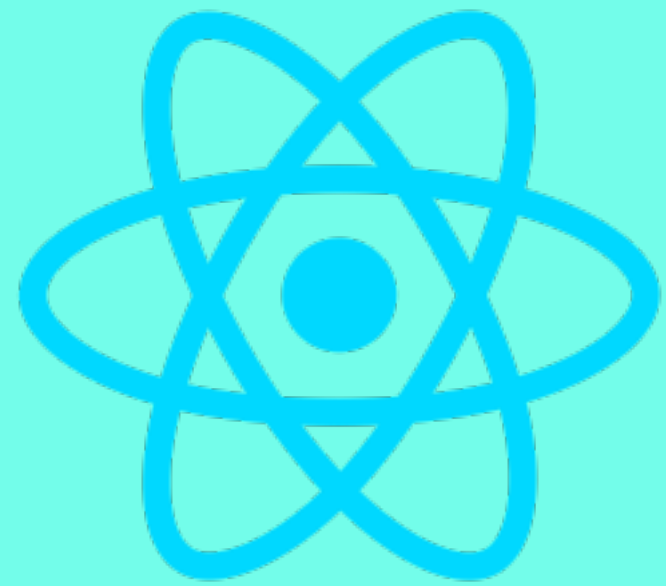


Permet l'accès au Store

Effectuer le dispatch d'une action

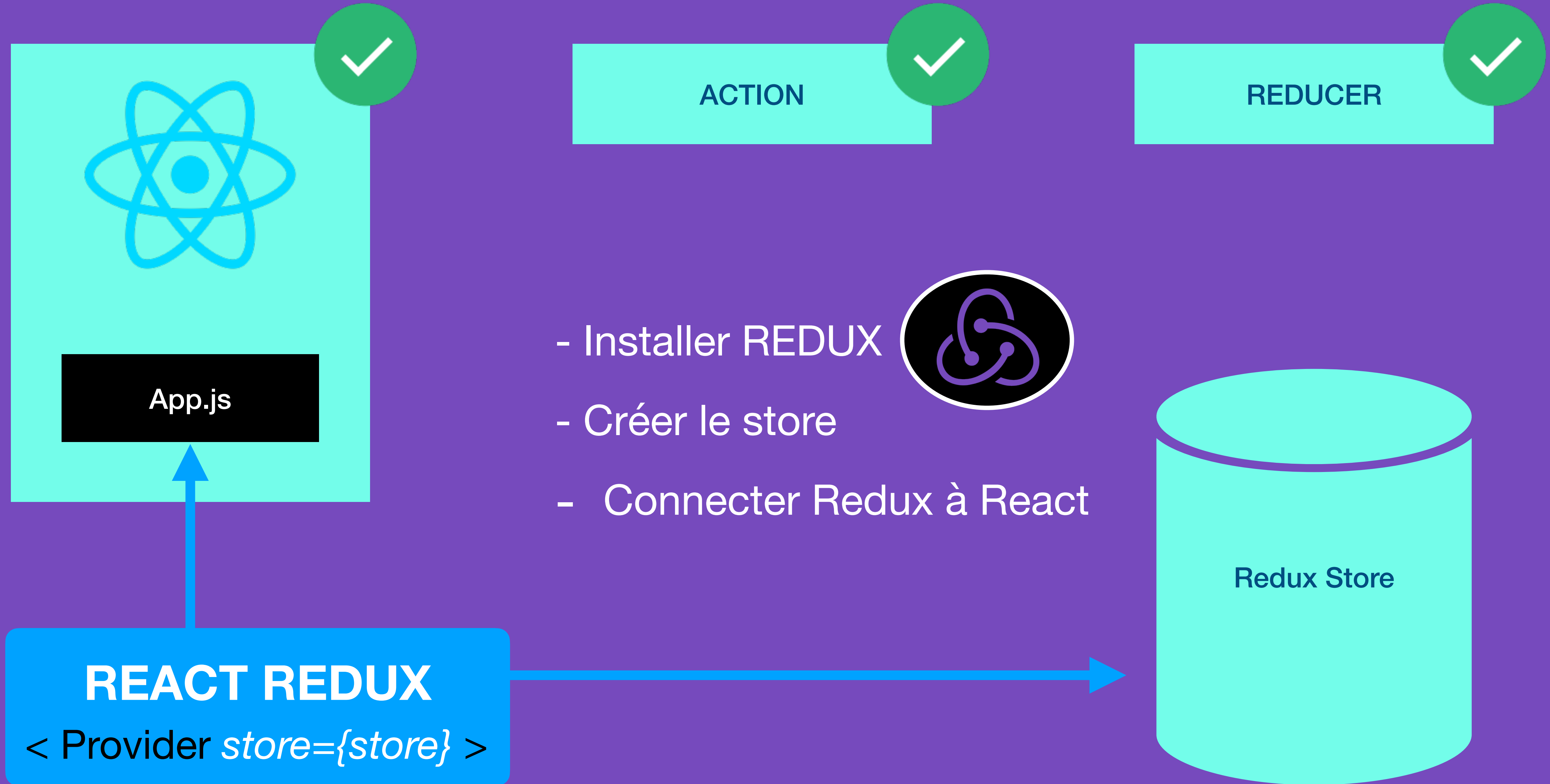
Le listener se lance à chaque modification du state (dans store)

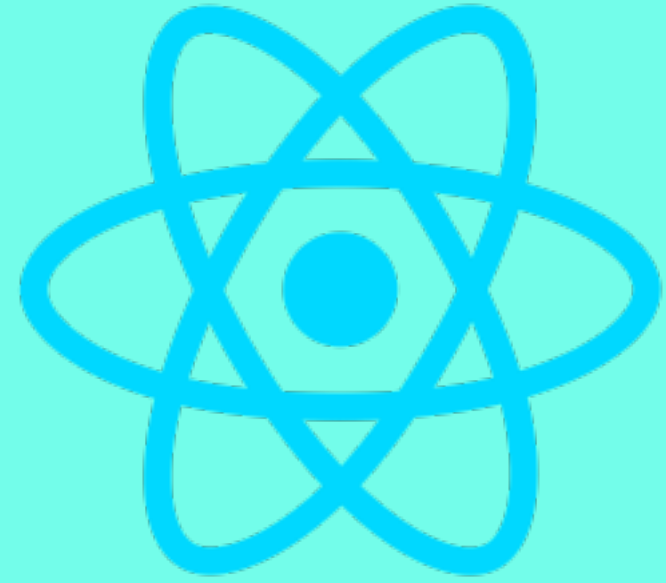
- Créer notre application React via create-react-app
- Créer notre composant **PhoneComponent**
- Créer notre **Action** via la fonction de création d'action
- Créer notre **Reducer**



ACTION

REDUCER





App.js

phoneContainer.js

mapStateToProps()
mapDispatchToProps()

useSelector (hook)
useDispatch (hook)

ACTION

REDUCER

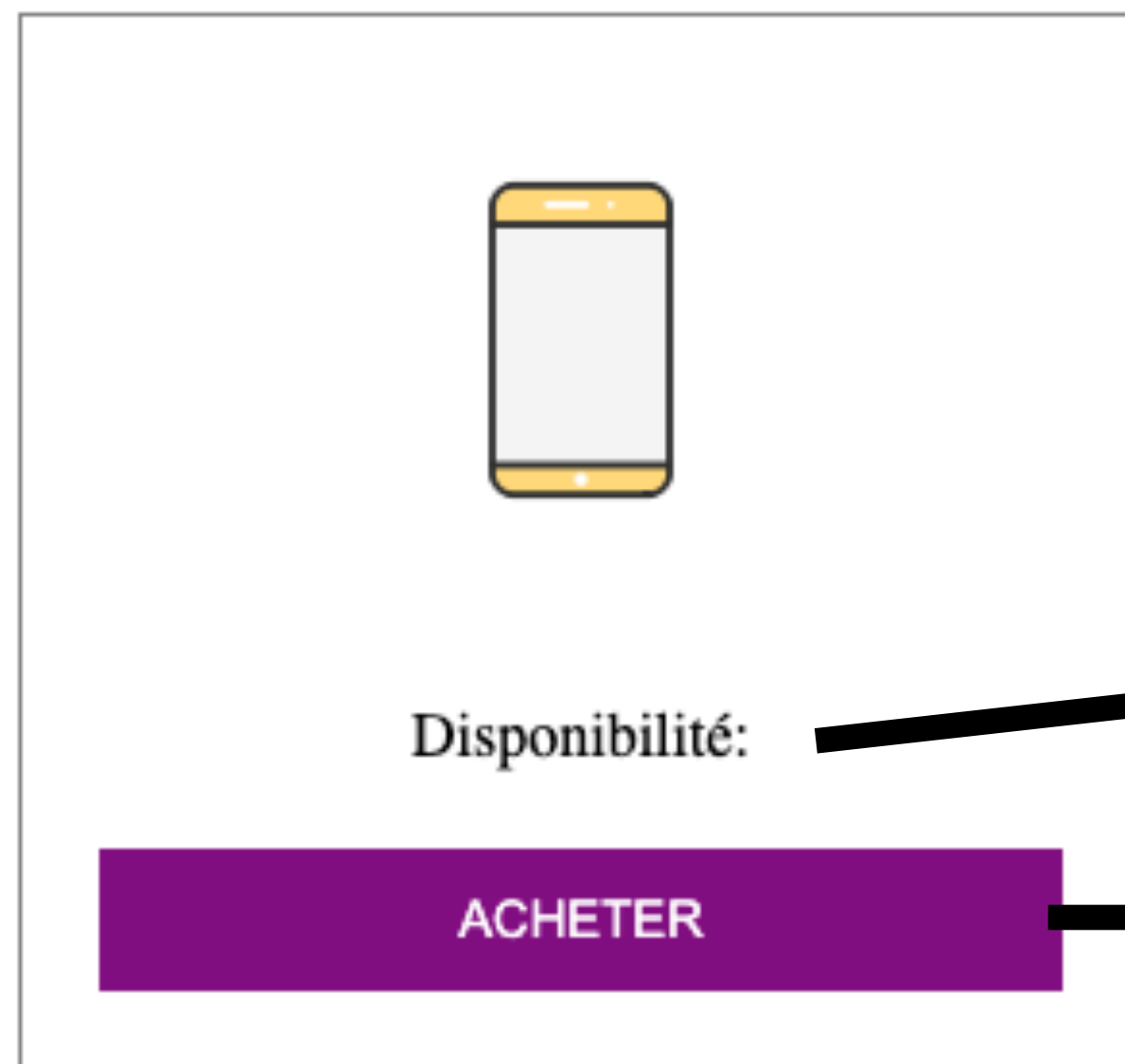
REACT REDUX

Provider
connect() HOC

Redux Store

phoneContainer.js

```
Function PhoneComponent( ) {
```



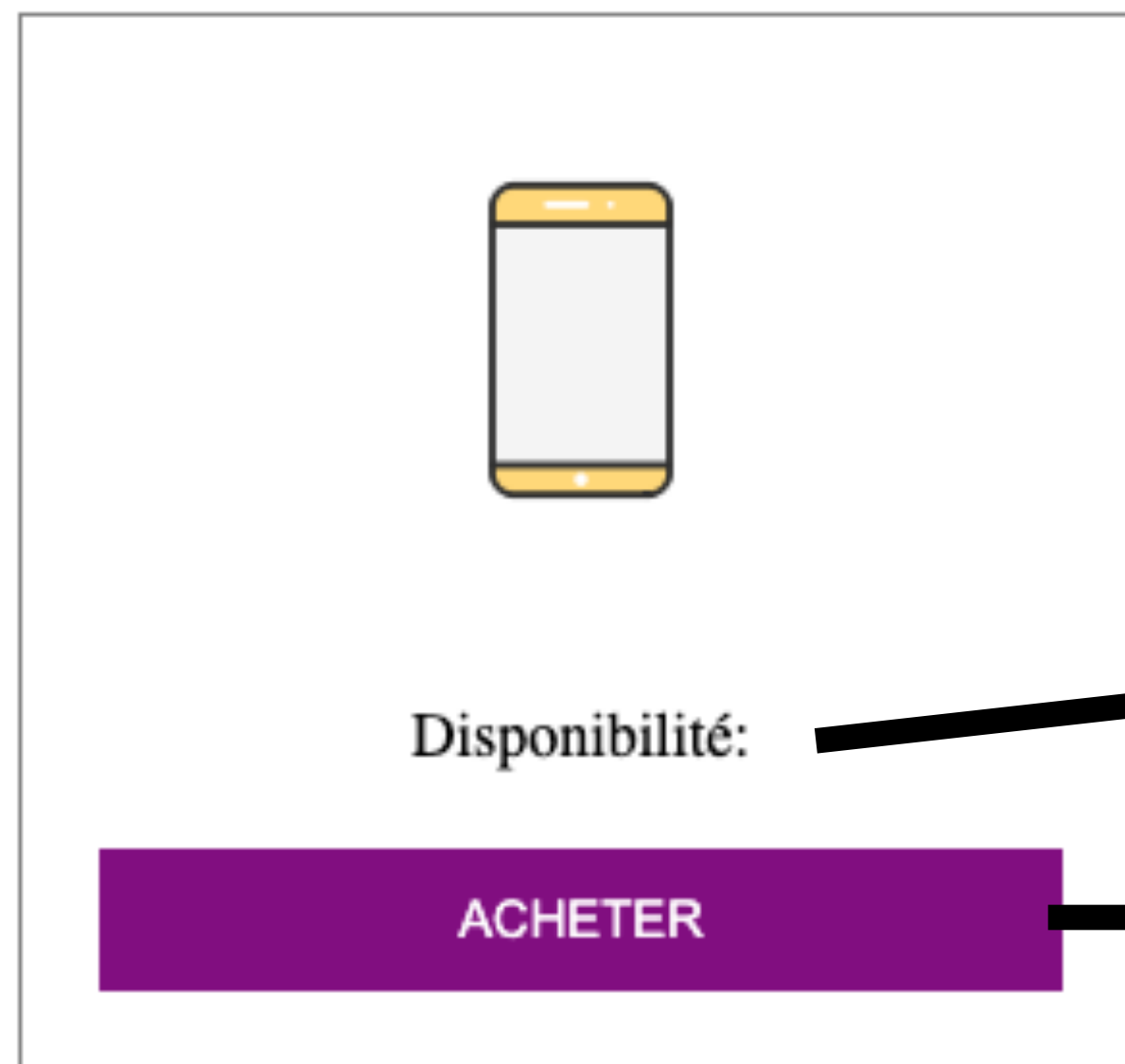
Obtenir le State

Dispatch(ACTION)

```
}
```

phoneContainer.js

```
Function PhoneComponent( props ) {
```



Obtenir le State

Dispatch(ACTION)

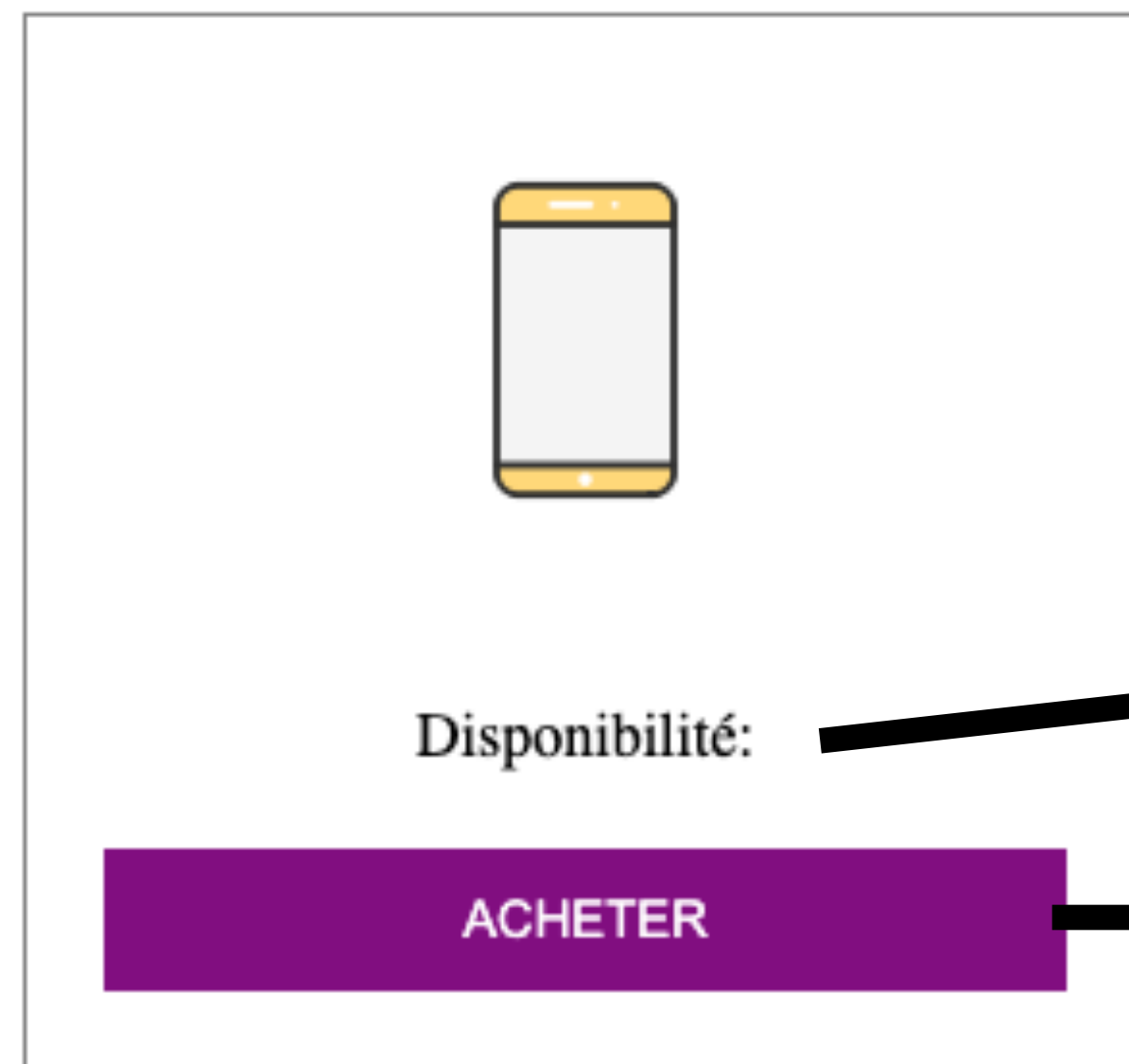
connect() HOC

mapStateToProps()
mapDispatchToProps()

```
}
```

phoneContainer.js

```
Function PhoneComponent( ) {
```



Obtenir le State

Dispatch(ACTION)

REACT REDUX



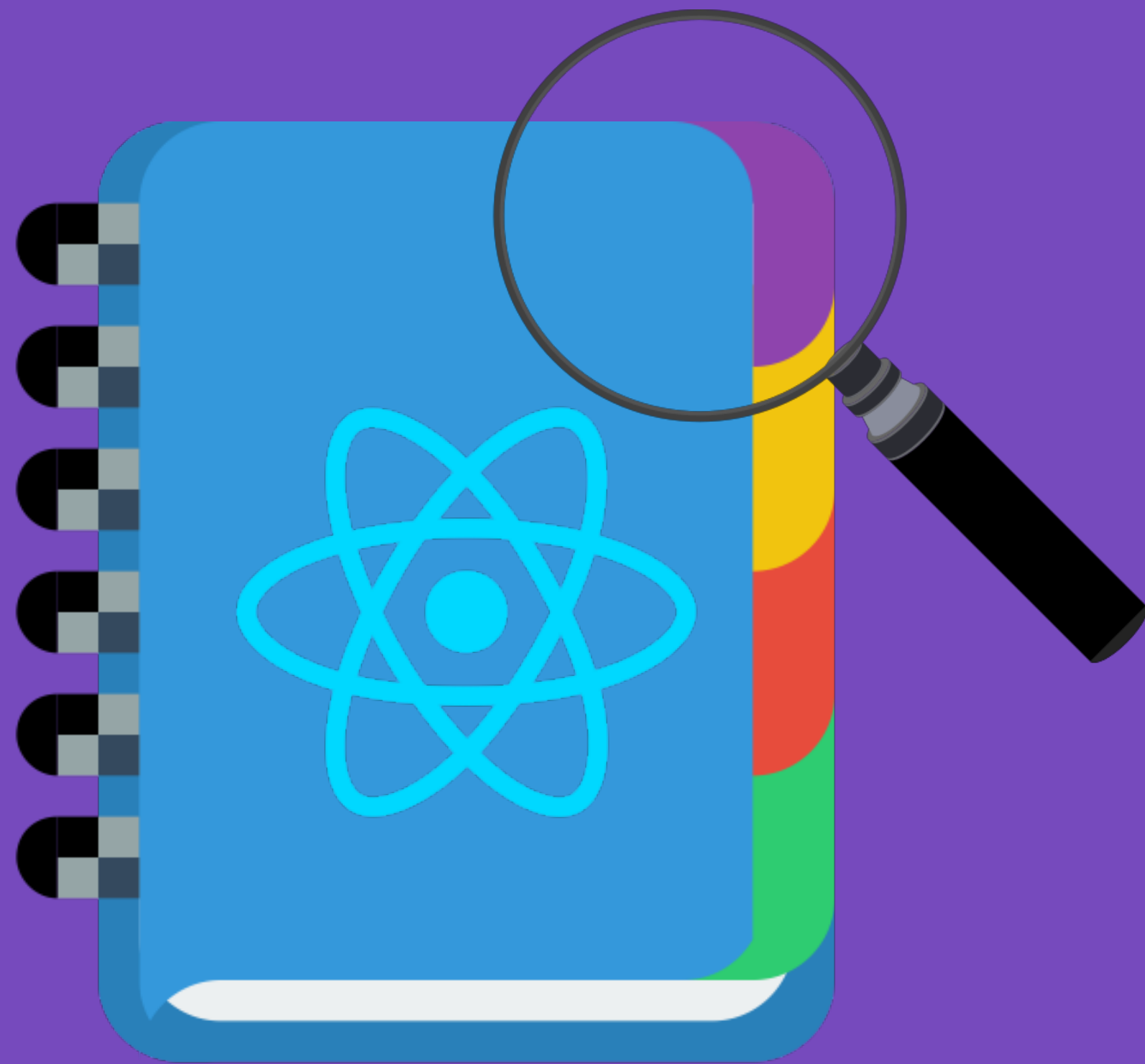
useSelector (hook)



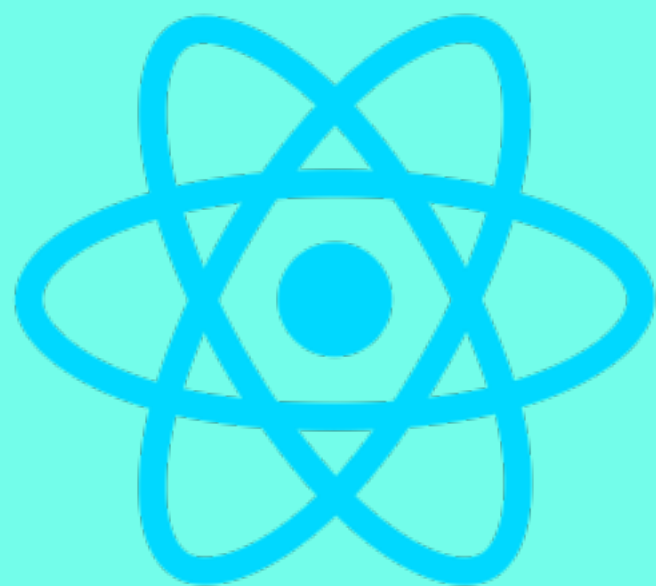
useDispatch (hook)

```
}
```


Lire la documentation



Les bonnes informations + mises à jour



App.js



phoneContainer.js



tvContainer.js

ACTION BUY_PHONE



ACTION BUY_TV



PhoneReducer



TvReducer



REACT REDUX



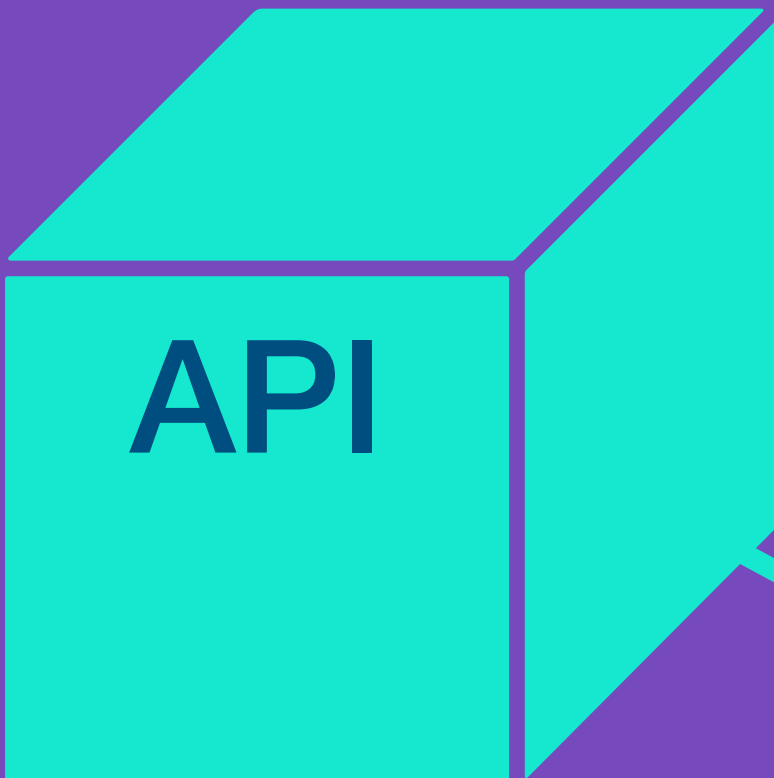
Provider
connect() HOC




Redux Store



jsonplaceholder






Disponibilité: **5**

ACHETER

1



Disponibilité: **10**

ACHETER

Commentaires

Comment 1

Comment 2

Function A()



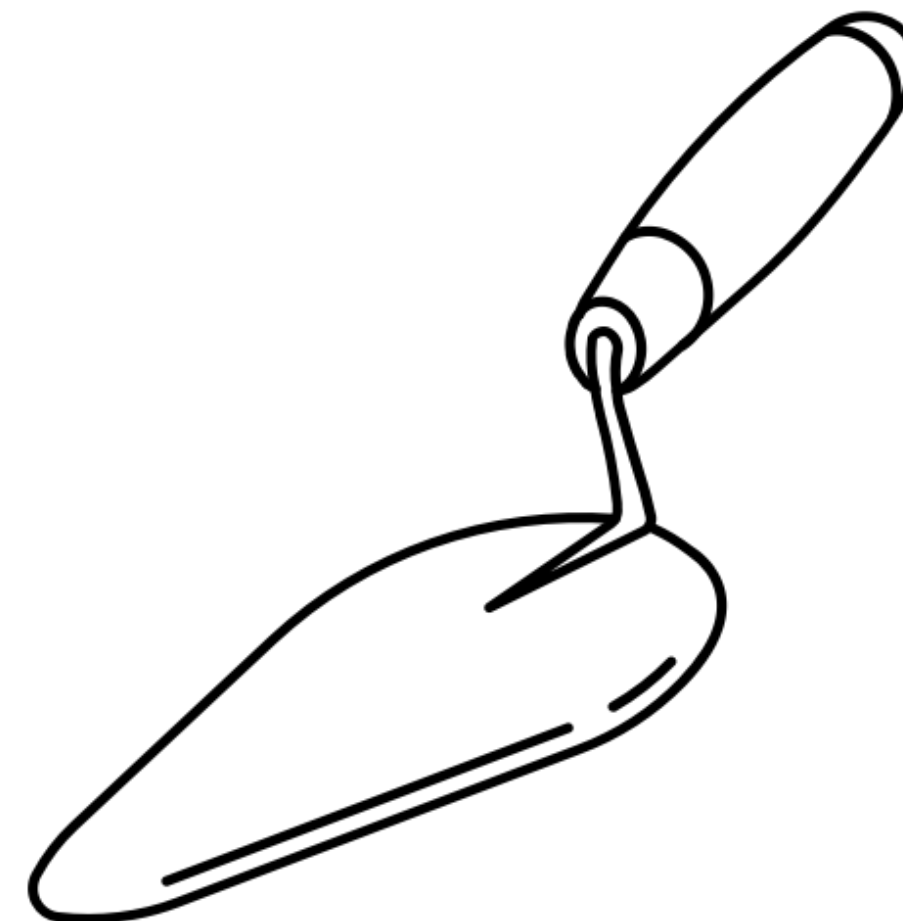
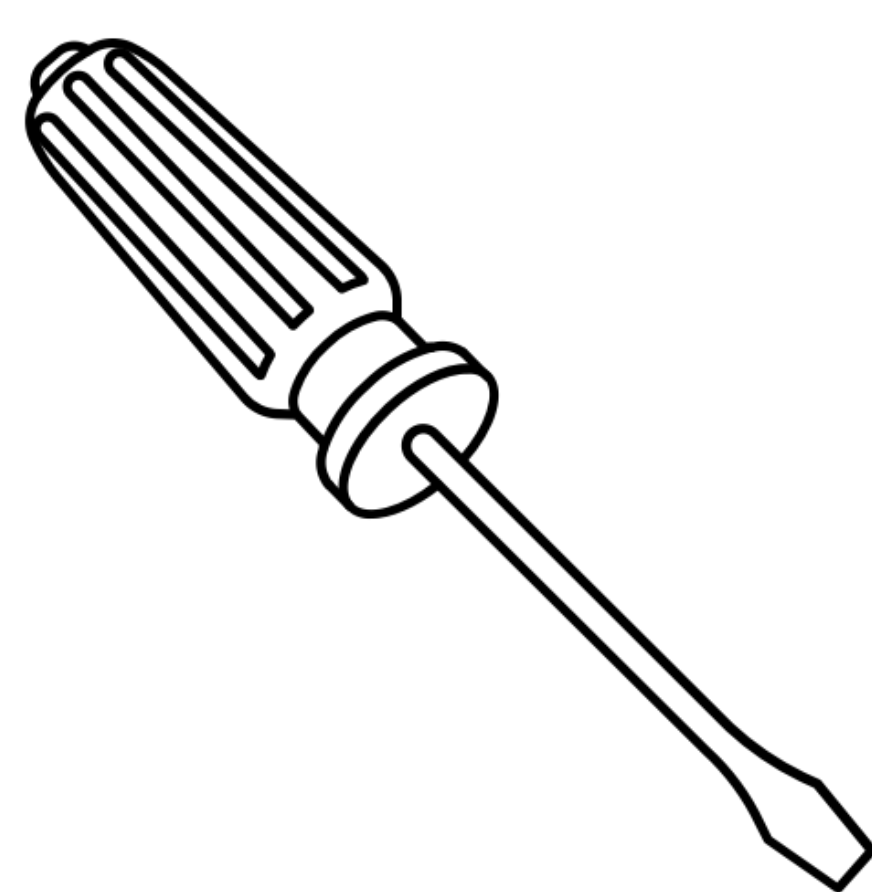
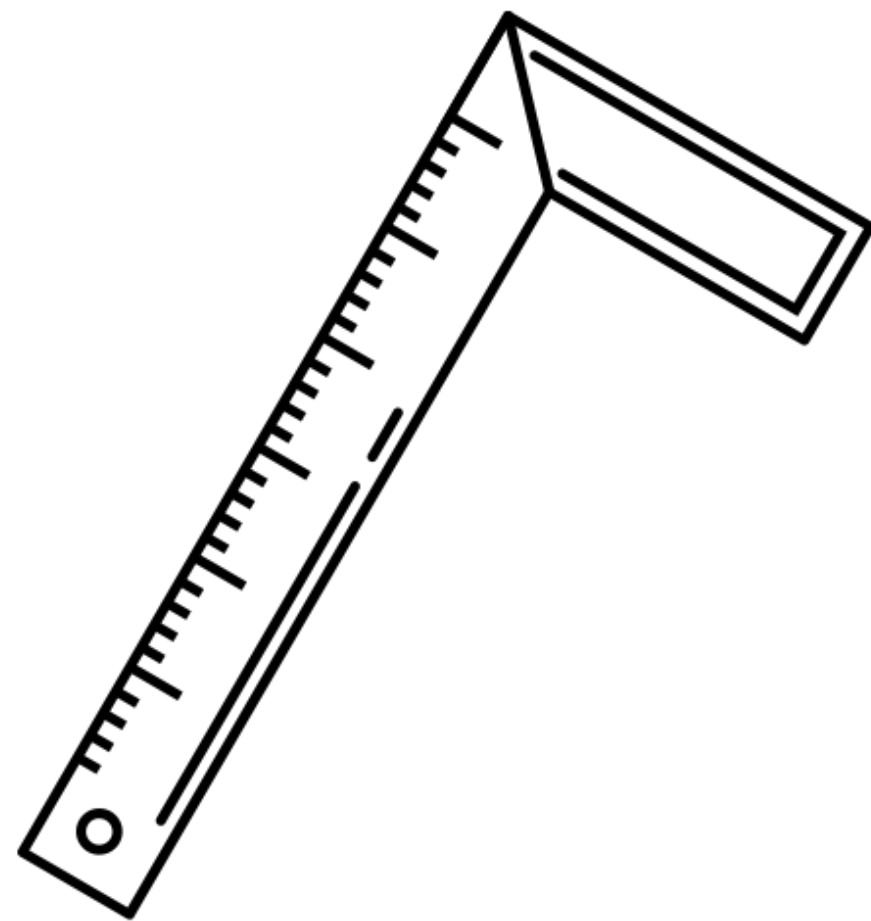
Middleware



Function B()

Redux Middleware

(plus de fonctionnalités)





createStore(reducer)



reducer1 reducer2
 ↓ ↓
combineReducers({ key: value, key: value })



rootReducer

applyMiddleware()



`createStore(reducer,)`



`combineReducers({ key: value, key: value })`

reducer1 reducer2



rootReducer

`applyMiddleware()`

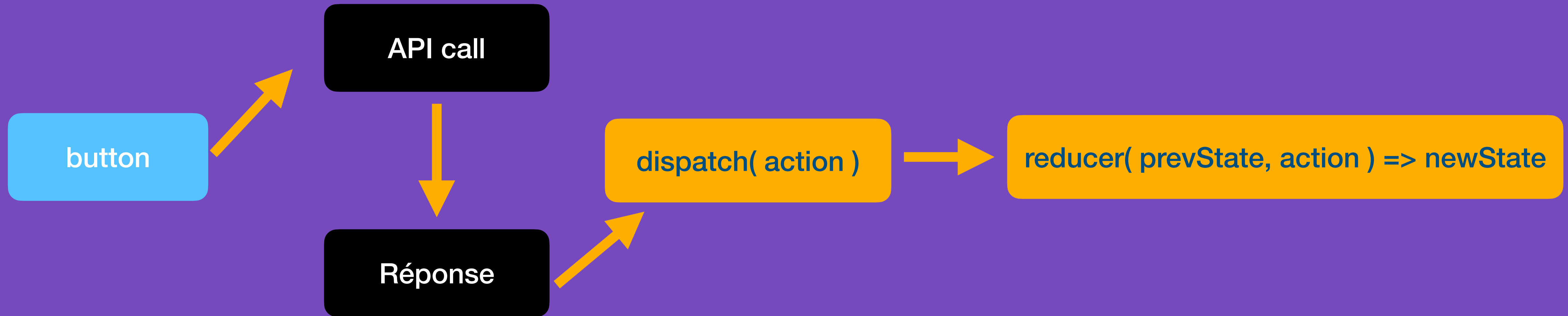
Redux-Thunk

`dispatch(action)` → (Middleware - API calls) → `reducer()`

1- Action synchrone



2 - Action asynchrone (exemple : API call)



Montage du
composant

Fonction asynchrone (middleware thunk pour appels API) ✓

dispatch(action)

dispatch(loadApiComments)

Réponse

dispatch(loadCommentsSuccess)

dispatch(loadCommentsError)

reducer(prevState, action) => newState

middleware

