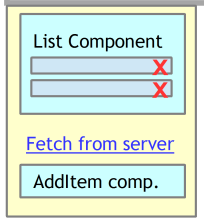
Components

```
var ListContainer = React.createClass({
  getInitialState: function(){
    return {  // Récup ETAT INITIAL depuis LE STORE
      list: todoStore.getList()    };
  },
  componentDidMount: function(){
    // Enreg. d'un LISTENER auprès du STORE
    todoStore.addChangeListener(this._onChange);
  },
  handleFetchRemote: function() {
    todoActions.fetchRemoteItem(..);
  },
  _onChange: function(){
    // Le LISTENER récup. le NOUVEL ETAT du STORE
    this.setState({  list: todoStore.getList()    });
  },
  render: function(){
    return (
     <a href="#" onClick={this.handleFetchRemote}>
          Fetch</a>
     <List items={this.state.list}  />
    ) }
});
```



Actions

```
var todoActions = {
  // Lister les Actions. Elles prep.
  // les data avt Dispatch.
  addItem: function(item){
    AppDispatcher.handleViewAction({
      actionType: appConstants.ADD_ITEM,
      data: item    });    },
  fetchRemoteItem: function(itemId) {
    Api.fetchRemoteItem(itemId);   }
};
```

API – Request Server
=> callback

```
var serverActions = {
  okResponse: function(…) {
    var serverAction = {
      ok: true,.. resultData: resultData };
    AppDispatcher.handleServerAction
                        (serverAction);
  },
```

Dispatcher

```
var AppDispatcher = new Dispatcher();

// Le Dispatcher est un Bridge
// - Les Actions le ref. pour envoyer un Event
// - Les Stores le référencent pour s'enregistrer et recevoir les Events.
AppDispatcher.handleViewAction =function(action){
  this.dispatch({
    // Attach this Source info
    source: VIEW,
    action: action
  });
};
AppDispatcher.handleServerAction =function(action)
{
  this.dispatch({
    // Attach this Source info
    source: SERVER,
    action: action
  });
};
```

Stores

```
// The MODEL -- Data maintained by this Store.
var _store = {  list: []   };

// The SETTERS - Private to this Module. Update Model Data.
var addItem = function(item){_store.list.push(item); };

// The STORE itself
var todoStore = objectAssign({}, EventEmitter.prototype, {
  addChangeListener: function(cb){
    this.on(CHANGE_EVENT, cb);
  },
  // The exposed GETTERS
  getList: function(){    return _store.list; },
});
```

```
// Registration with AppDispatcher
AppDispatcher.register(function(payload){
  if(payload.source === VIEW){
    switch(payload.action.actionType){
      case appConstants.ADD_ITEM:
        addItem(action.data);     break;     …… }
  } else if(payload.source === SERVER){
    if(payload.action.ok){
      addItem(payload.action.resultData.body);
    } else {     addItem("Err: " + payload.action.errCode);   }
  } else {    return true;  }

  // ESSENTIAL
  todoStore.emit(CHANGE_EVENT);
});
```