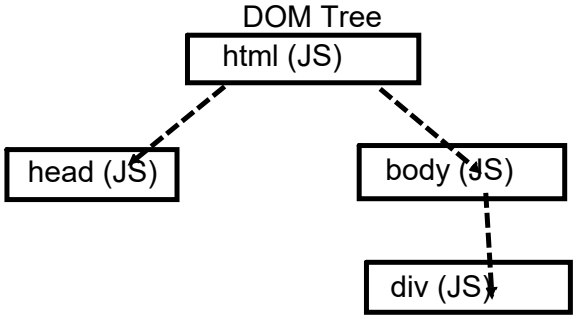
React JS : V of MVC

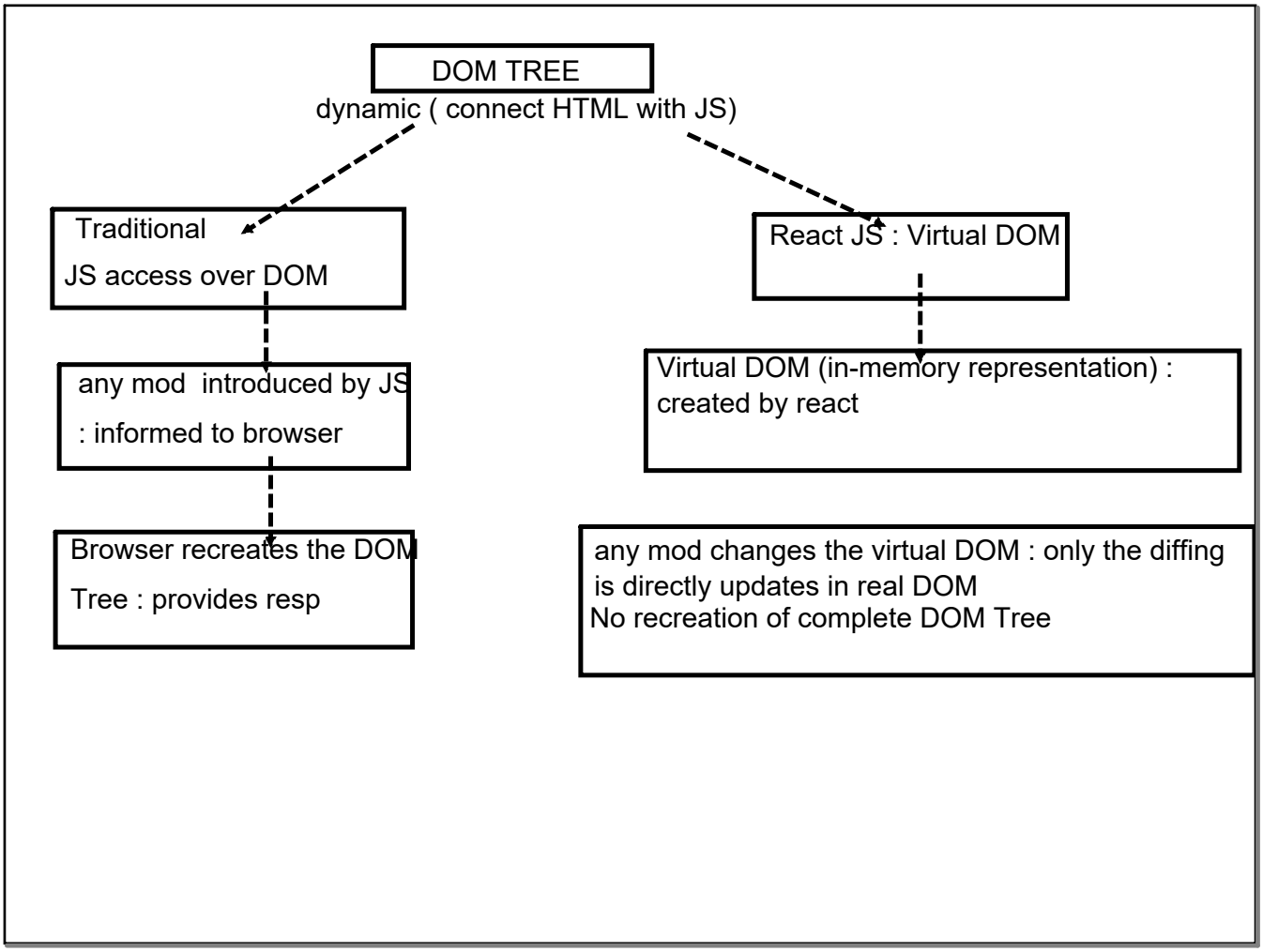
Angular : framework : MVC/W

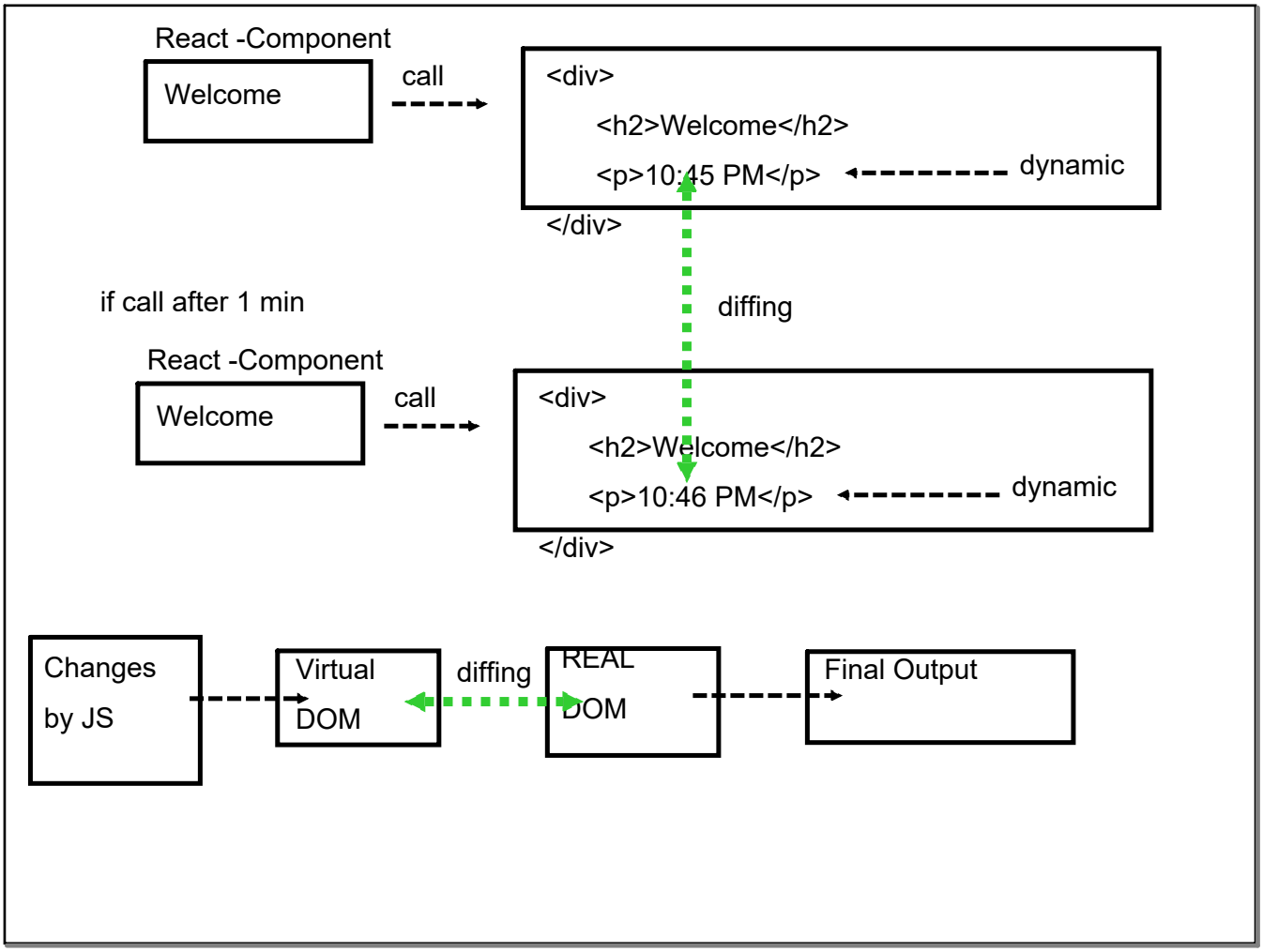


DOM : Document Object Model

Every HTML Component is converted into JS Object

```
<html>
    <head></head>
    <body>
        <div></div>
    </body>
</html>
```


DOM TREE
dynamic (connect HTML with JS)
Traditional
JS access over DOM
any mod introduced by JS
: informed to browser
Browser recreates the DOM
Tree : provides resp
React JS : Virtual DOM
Virtual DOM (in-memory representation) :
created by react
any mod changes the virtual DOM : only the diffing
is directly updates in real DOM
No recreation of complete DOM Tree


React -Component
Welcome
call

<h2>Welcome</h2>
10:45 PM
dynamic

if call after 1 min
diffing
React -Component
Welcome
call

<h2>Welcome</h2>
10:46 PM
dynamic

Changes by JS
Virtual DOM
diffing
REAL DOM
Final Output

React : Component base programming

==> reusable HTML Comp (backed by JS)

Resources : react js lib (engine): available as single file (eg: jquery)

additional lib : react-dom.js (virtual DOM)

backend/programming : Domain specific language : JSX (ES6)

=>javascript and XML

=> ES6(not supported by browser)

Transpiler : one S/C structure to another S/C structure

JSX + ES6 ----------> (ES5)

babel/TRaceur

IDE : VS Code

node installation

npm :

need to install : react CLI

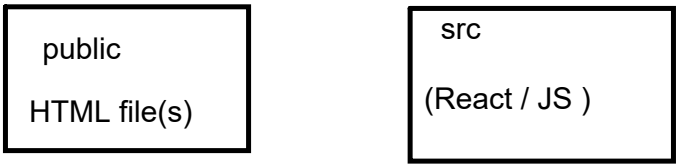
Facebook : React-app boilerplate generator (create-react-app : need to install)

installing react-tool

>npm i -g create-react-app

creating a react app:

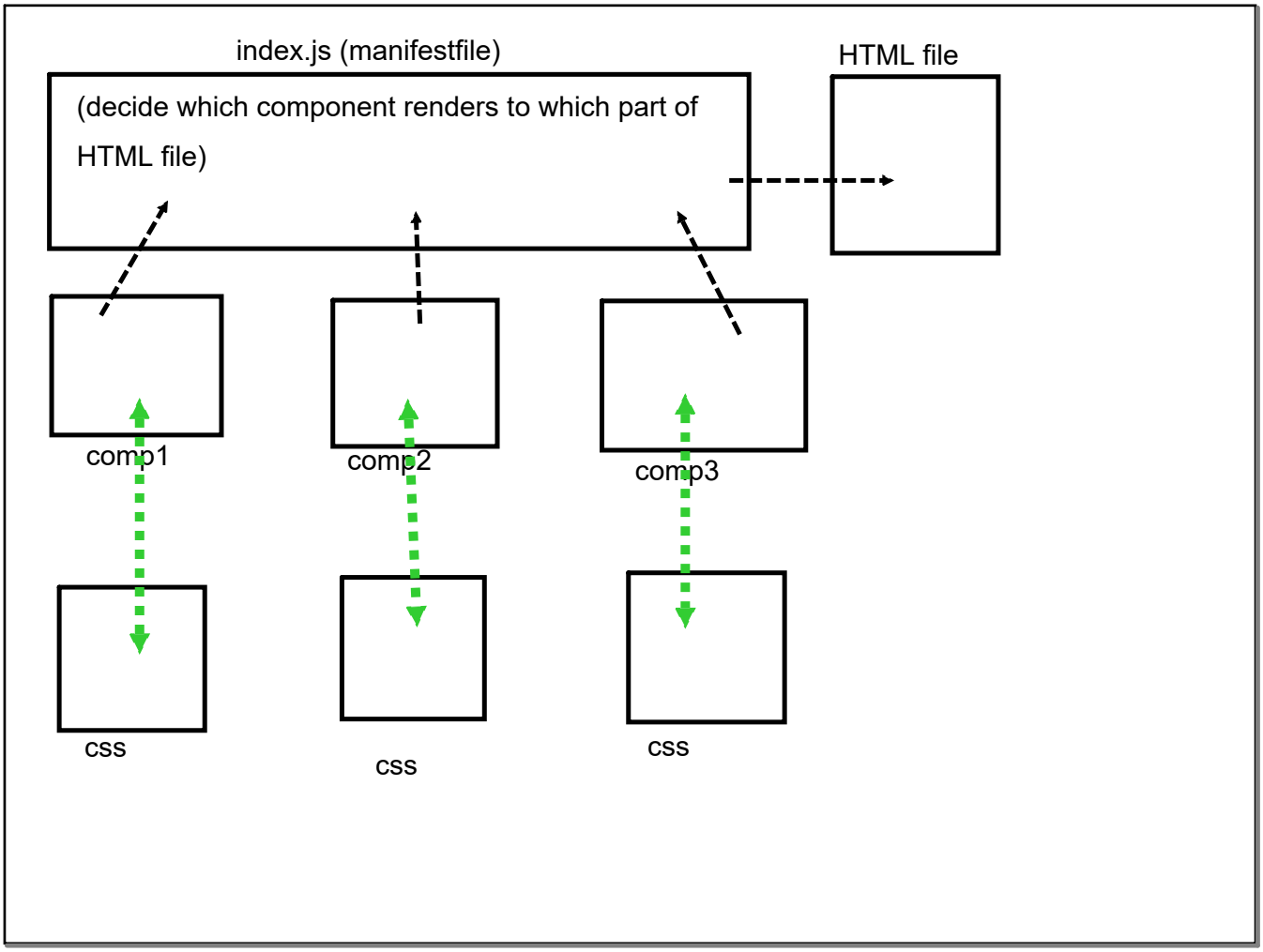
>create-react-app <project-name>



HTML file will use a JS(ES5) converted from React Resources

3 specific resources for a React comp

1. js file (contains code for React comp)
2. test file (unit test code for that comp)
3. css file (css for that component)


index.js (manifestfile)
(decide which component renders to which part of HTML file)
HTML file
comp1
comp2
comp3
css
css
css

Component named as APP : new tag <app> (not a HTML tag)

(JSX TAG)

Launching React app : npm start

Creating React Component:

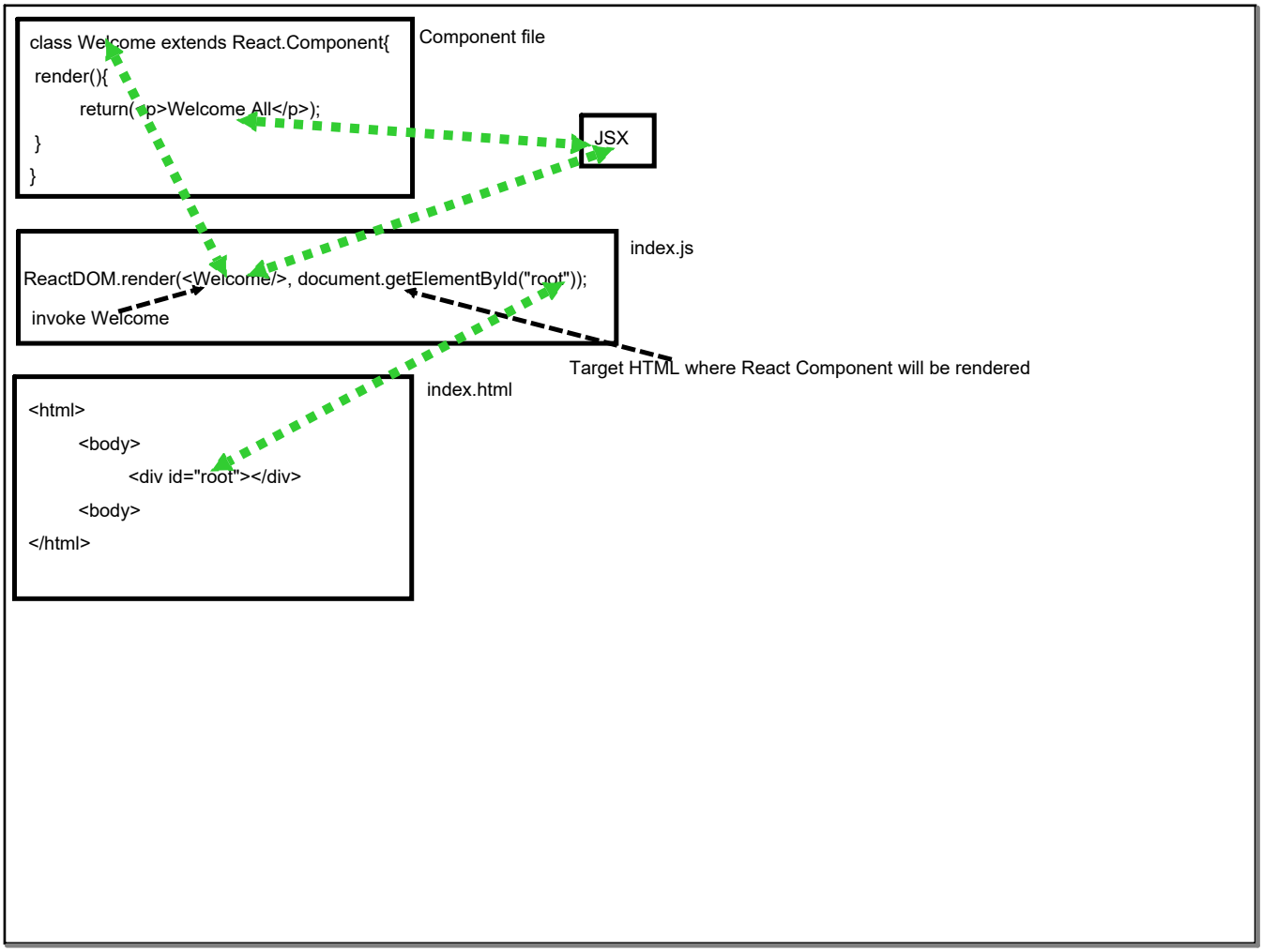
React comp : JS class

Create a JS class : extends (inherit) React.Component

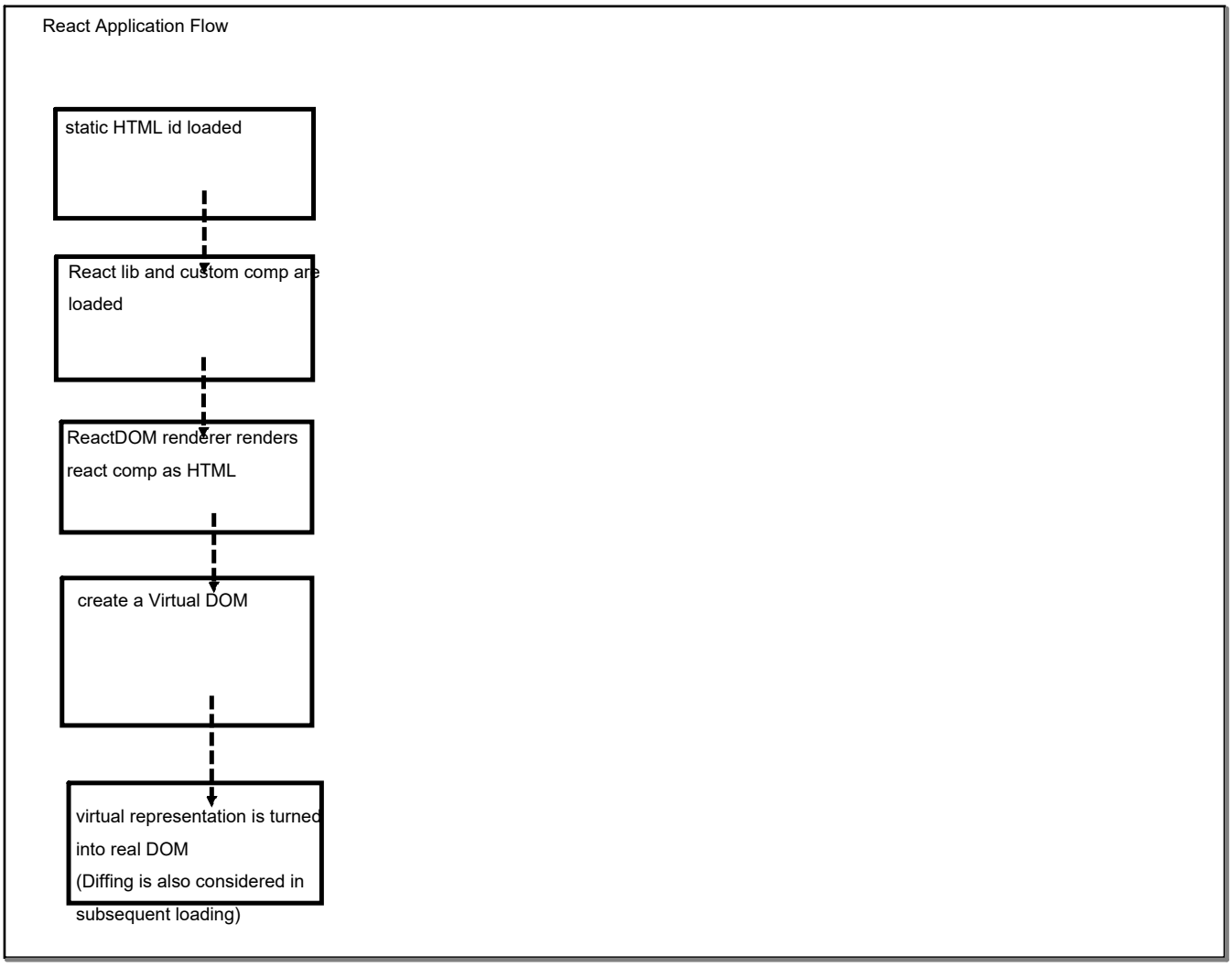
```
class Welcome extends React.Component{
    // method needs to override
    render(){
        // must return the dynamic UI to render
        return (<JSX Code>)
    }
}
```

need to specify where to put output (index.js)

```
ReactDom.render(<component name>, <location on html web
page where to provide its output>);
```


class Welcome extends React.Component{
render(){
return(Welcome All);
}
}
Component file
JSX
ReactDOM.render(<Welcome/>, document.getElementById("root"));
invoke Welcome
index.js
Target HTML where React Component will be rendered
<html>
<body>

<body>
</html>
index.html


React Application Flow
static HTML id loaded
React lib and custom comp are loaded
ReactDOM renderer renders react comp as HTML
create a Virtual DOM
virtual representation is turned into real DOM
(Diffing is also considered in subsequent loading)

JSX (Javascript XML) : (XML in background maintaining relationship beteen HTML and JS)

JSX : another way of writing JS with transpile step

```
class Welcome extends React.Component{
 render(){
        return <p>Welcome All</p>;
 }
}
```

React.createElement('p', null, 'Welcome All')

```
ReactDOM.render <Welcome/> document.getElementById("root"));
```

React.createElement(Welcome, null)

createElement : JS method to create a HTML Comp on the fly

```
<div>
        <h3>Welcome All</h3>
        <p style='color:RED'>Sample para</p>
</div>
```

```
React.createElement('div', null,
        React.createElement('h3', null, 'Welcome All'),
        React.createElement('p', {style: 'color:RED' }, 'Sample para') );
```

While writing JSX : take care of using JS keyword

```
React.createElement('div', null,
        React.createElement('h3', null, 'Welcome All'),
        React.createElement('p', {className : 'some-name' }, 'Sample para') );
```

Generated HTML

```
<div>
        <h3>Welcome All</h3>
        <p class='some-name'>Sample para</p>
</div>
```

Writing pure JS code in JSX : binding in curly set of braces

{Javascript}

Use Case :

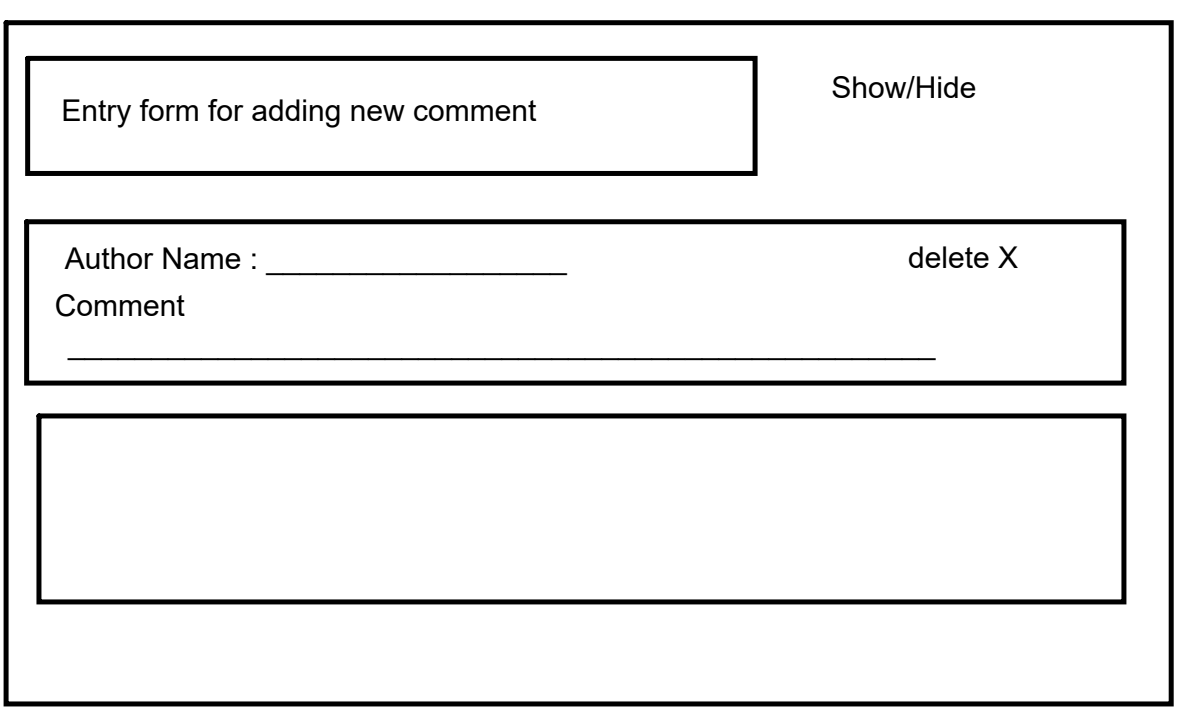
Commenting Engine:

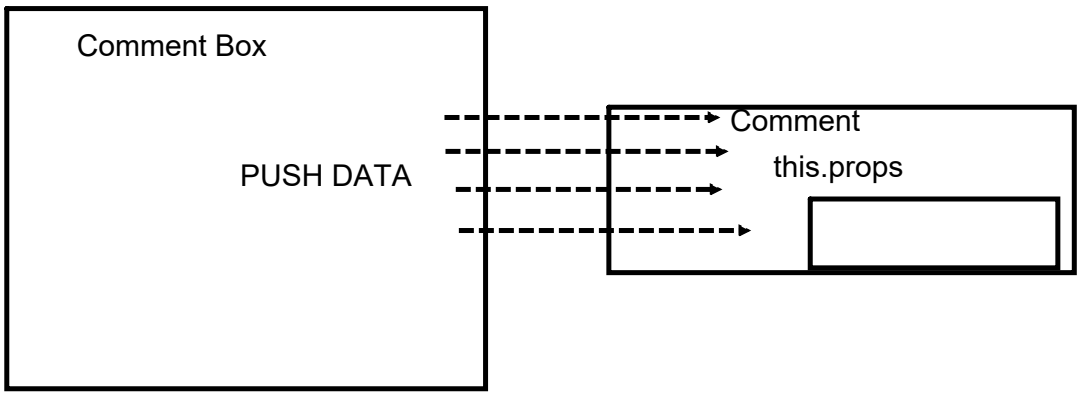
==> allow visitors to post comments on blog post...

==> able to show and hide comments

==>Able to add new comment

==>able to delete comment





need to pass values to Comment comp from Comment Box (Arguments)

Arguments passed as 'props'

Usage : looks like attributes of HTML

Reading props

All props passed is available in a inbuilt object (this.props)

Type of prop values:

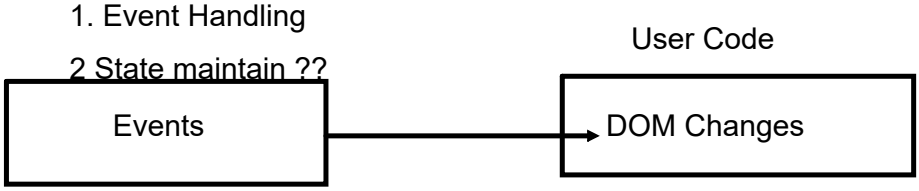
1. String
2. Number
3. Boolean
4. Array
5. Object
6. function

React JS by default provides one way data flow

Parent to child

1. Event Handling

2 State maintain ??



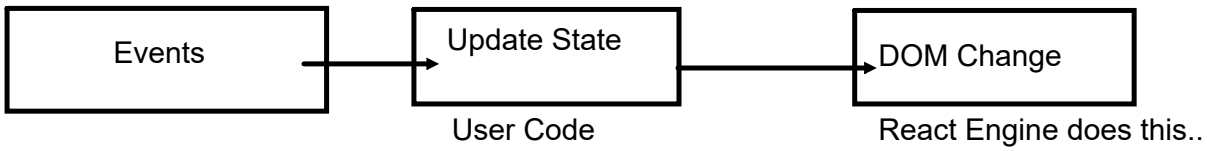
Show/Hide : Manipulation of DOM

1. Direct Manipulation (tradition , js/jquery)
2. Indirect DOM Manipulation (React : Virtual DOM)

Direct DOM Manipulation : eg: jquery

```
$('#show_hide_btn').on('click', function(){
    $('#comment_list').show();
    $('#comment_list').hide();
});
```

Manually Manipulation
(Re-renderring of complete DOM Tree)



=> We don't modify DOM directly

=>We modify the component state object (in response to event etc..)

=>React handles the update

State Object : JS Object live inside each Component (this.state)

: any modification in state object will trigger render() (callback)

render calling()

re-creates Virtual DOM

React Engine finds the diffing between real DOM and new Virtual DOM

Injects the changes only to real DOM

state : JS Object (by default empty)

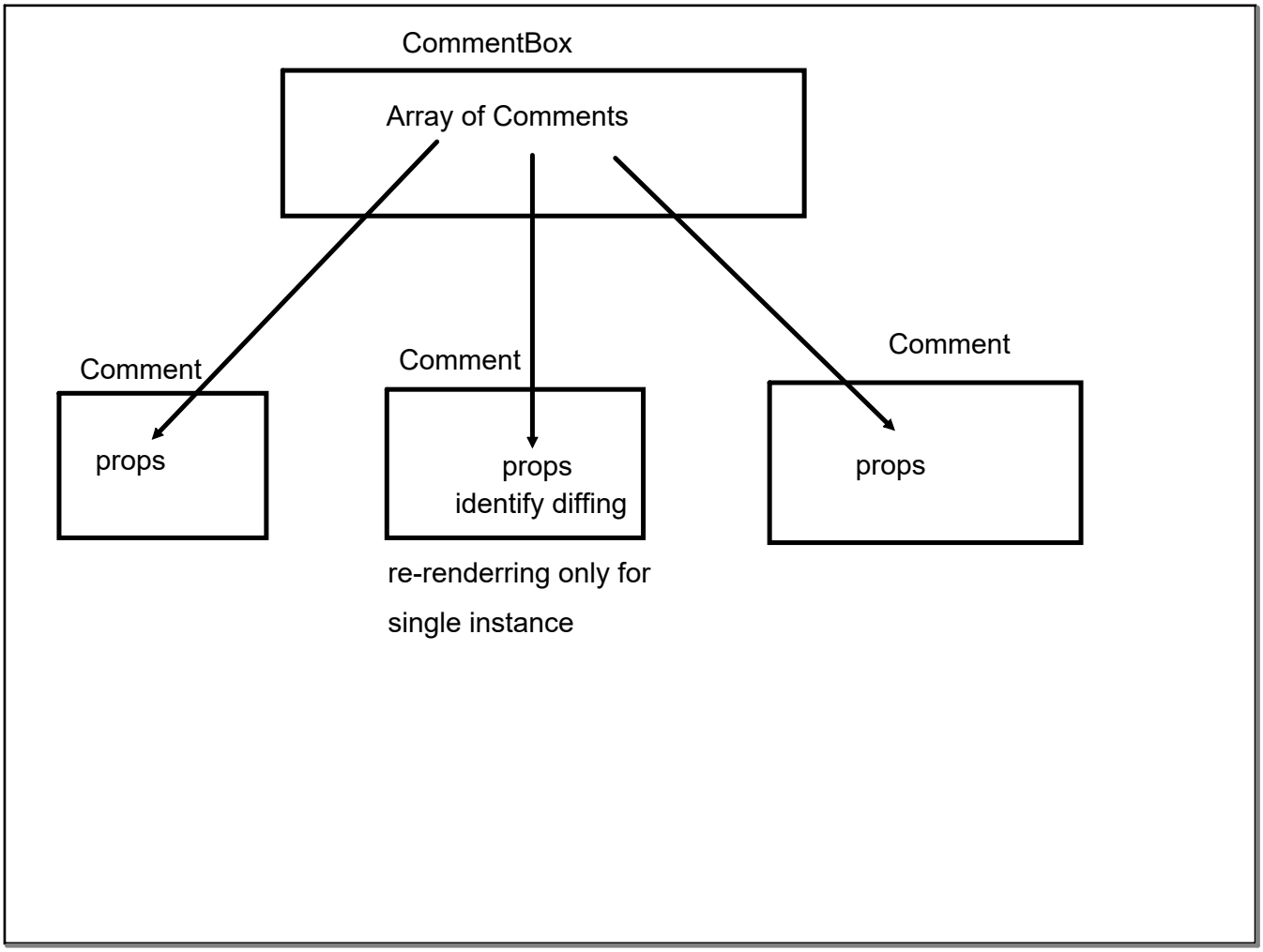
==> we need to add custom fld

==> constructor : set the initial state

Trigger of render takes place in following cases...

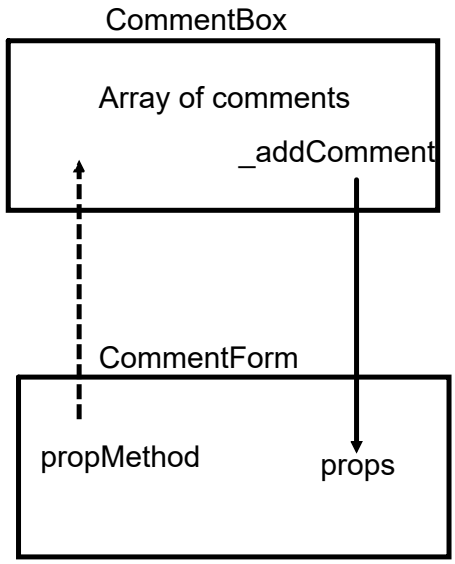
1. Changes in state object
2. any modification in props values
3. any modification in instance members

Note : Do not modify any of the above elements inside the render method (else could result into circular calls over render method)


CommentBox
Array of Comments
Comment
props
Comment
props
identify diffing
Comment
props
re-renderring only for
single instance

Adding new Comments :

need to add new component to add comments



_addComment of parent will be called by child component

==> Need to transfer comment list to state object

==> Need to have a custom method to receive/accept new info from child component and i update the state

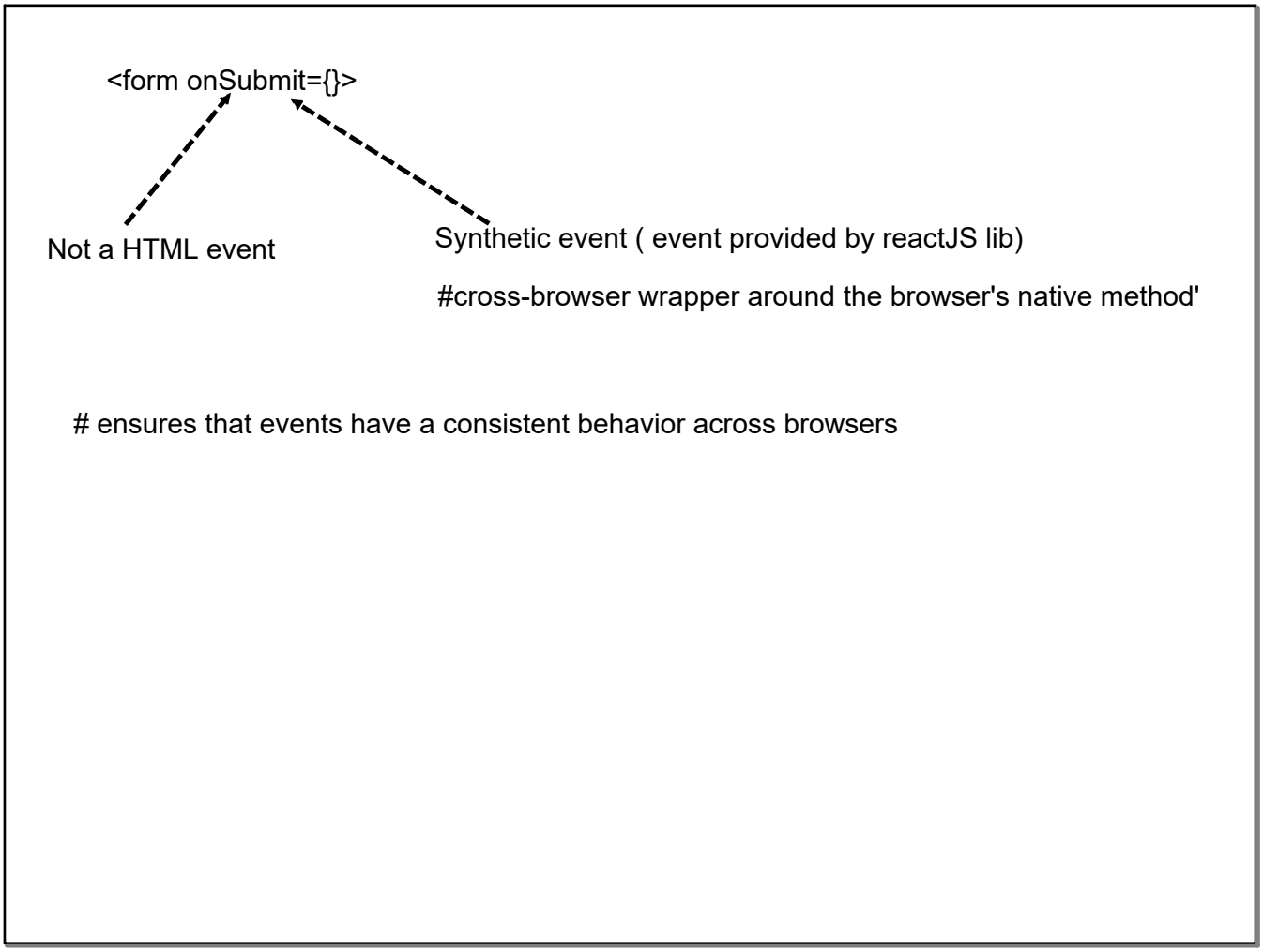
==> Need to have the values of form fields available as instance member of Component class (so that we can decide upon what to do with form fields)

ref (JSX attribute) : asks for a method to decide upon what to do with current html form field

```
<input placeholder="Author:"
    ref={(input) => this._author = input}/>
```

a new instance member would be created

==> Need to prevent the default behavior of Submit button


<form onSubmit={}>
Not a HTML event
Synthetic event (event provided by reactJS lib)
#cross-browser wrapper around the browser's native method'
ensures that events have a consistent behavior across browsers

Need to talk with server:

All interactions with server would be async (simple : jquery ajax)

1. fetch list of comments from server
2. new comment needs to shared with server
3. deletion to be informed to server
4. multi user env

React Component life cycle

constructor()
componentWillMount() : call _fetchComment (once) : before first render call
render() : multiple times
componentDidMount(); : after first renderring (once)
componentWillUnmount(); just before component is removed

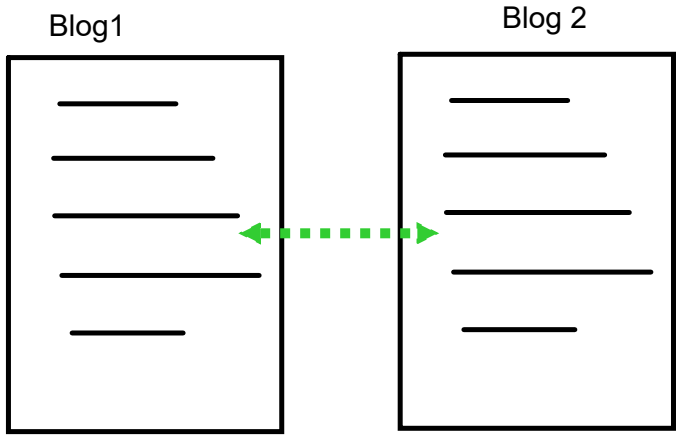
polling : periodically check for updates : continous call of fetchComment() at regular interval
Need to initiate a polling process : componentDidMount()

Lots of component

there will be continous change in component

- Adding and removal of component
- Routing between diff comp
- base data of component changed

Memory leaks : CommentBox will keep on loading data of previous blogs



1. support of jquery : need to add/install jquery lib : > npm add jquery --save
2. need to have a remote server

    json-server : fake REST API

install the json-server (npm -i -g json-server)

need to have a backend data:

    #create a json file (hold the data)

launch the json-server (need to tell json-server which json file to be used)

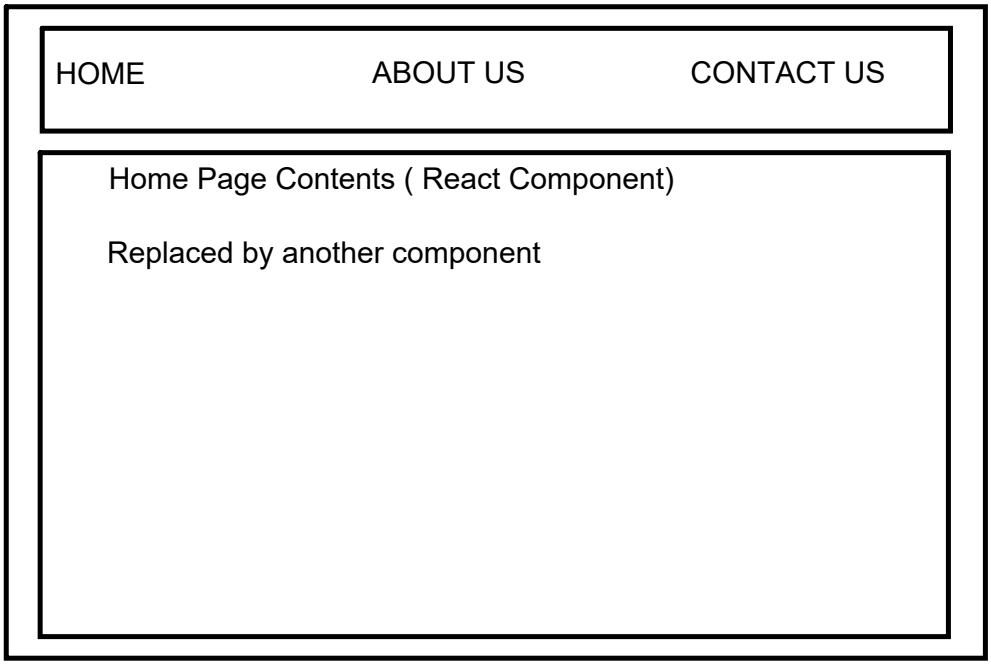
    > json-server (-p 2244) --watch db.json (by default : 3000)

json-server will activate std REST endpoint:

    http://localhost:3000/comments : (All http verbs)

http://localhost:3000/comments/1

SPA : Routing


HOME
ABOUT US
CONTACT US
Home Page Contents (React Component)
Replaced by another component

React Component

- # Home (default)
- # ContactUs
- # AboutUs

by default anchor (href) request the remote server and ask for a page mapped with url

need to map url with a react component

- #create a react component (Route)
- => map url with component
- => render that component

pass 2 prop values:

1. path/url to match
2. component to render when location matches path/url

JS specs

DOM Objects : HTML tags represented as JS Object

BOM Objects : JS Object to access the browser resources

#window

#history

#navigator etc

Route : would be responsible for renderring components based on the link used..

Two props:

1. path to match
2. component to render

Anchor (href) tag , reloads the page : sends the request for server

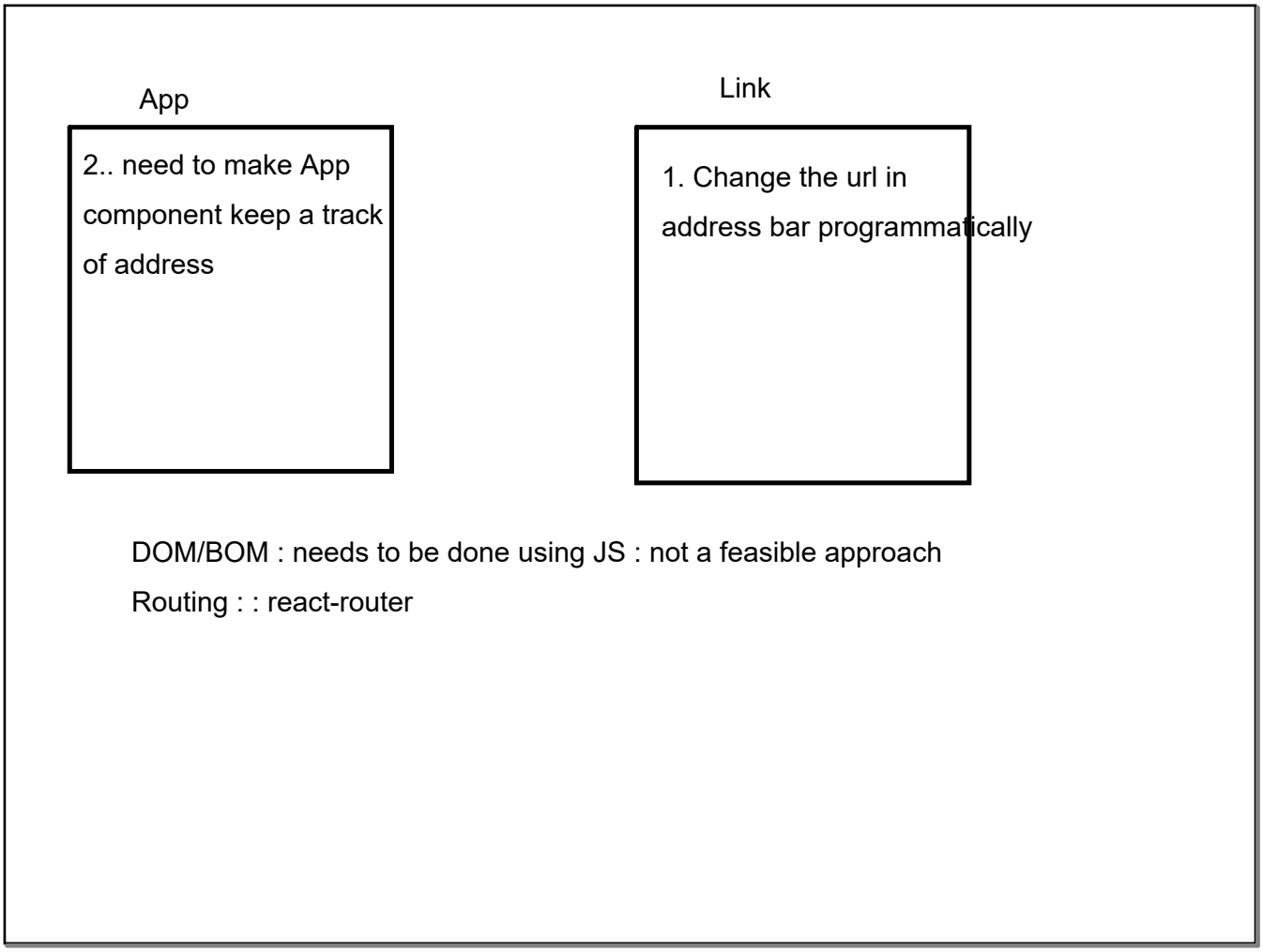
need to replace the anchor tag with custom tag

=> Add custom react component

=> wrap the anchor and customize the behavior

React component can access the tag body contents

this.props.children



DOM/BOM : needs to be done using JS : not a feasible approach

Routing : : react-router

Special library : react-router-dom

install the library

```
> npm install react-router-dom
```

contains pre-defined React Components

Route : works same as our custom component

Link : works same as expected

BrowserRouter : listens to changes in address bar

Need to add a 404 reponse (Not found)

Switch : renders component only when path matches

==>Allows to have a fallback component

Need to create a NotFound Component

Users

user1

user2

user3

Need to pass param with url

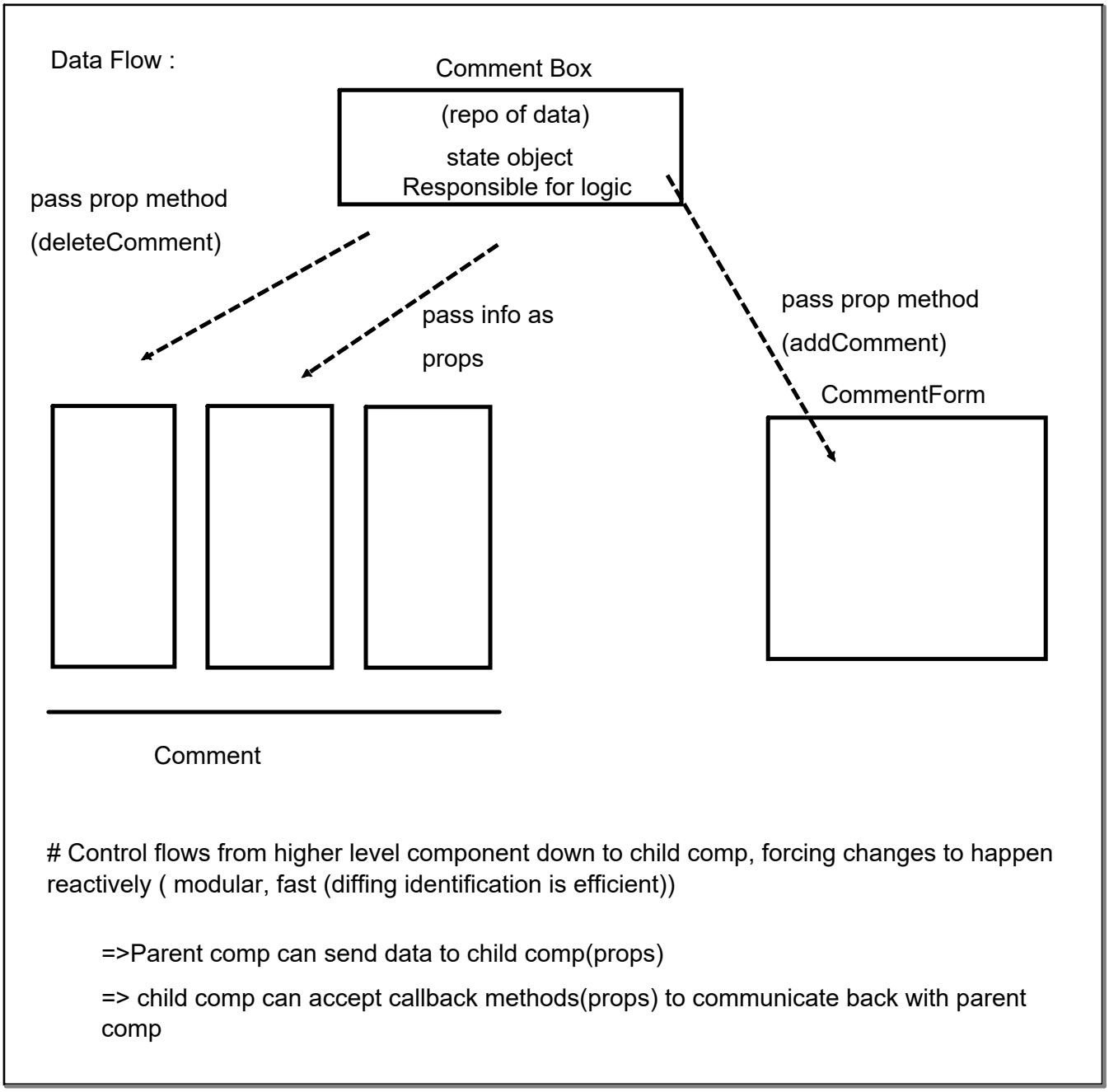
Need to access/receive in component

Navigate programmatically :

==> Every component is provided a history field in props object

=> history field : interacts internally wiht BOM objects

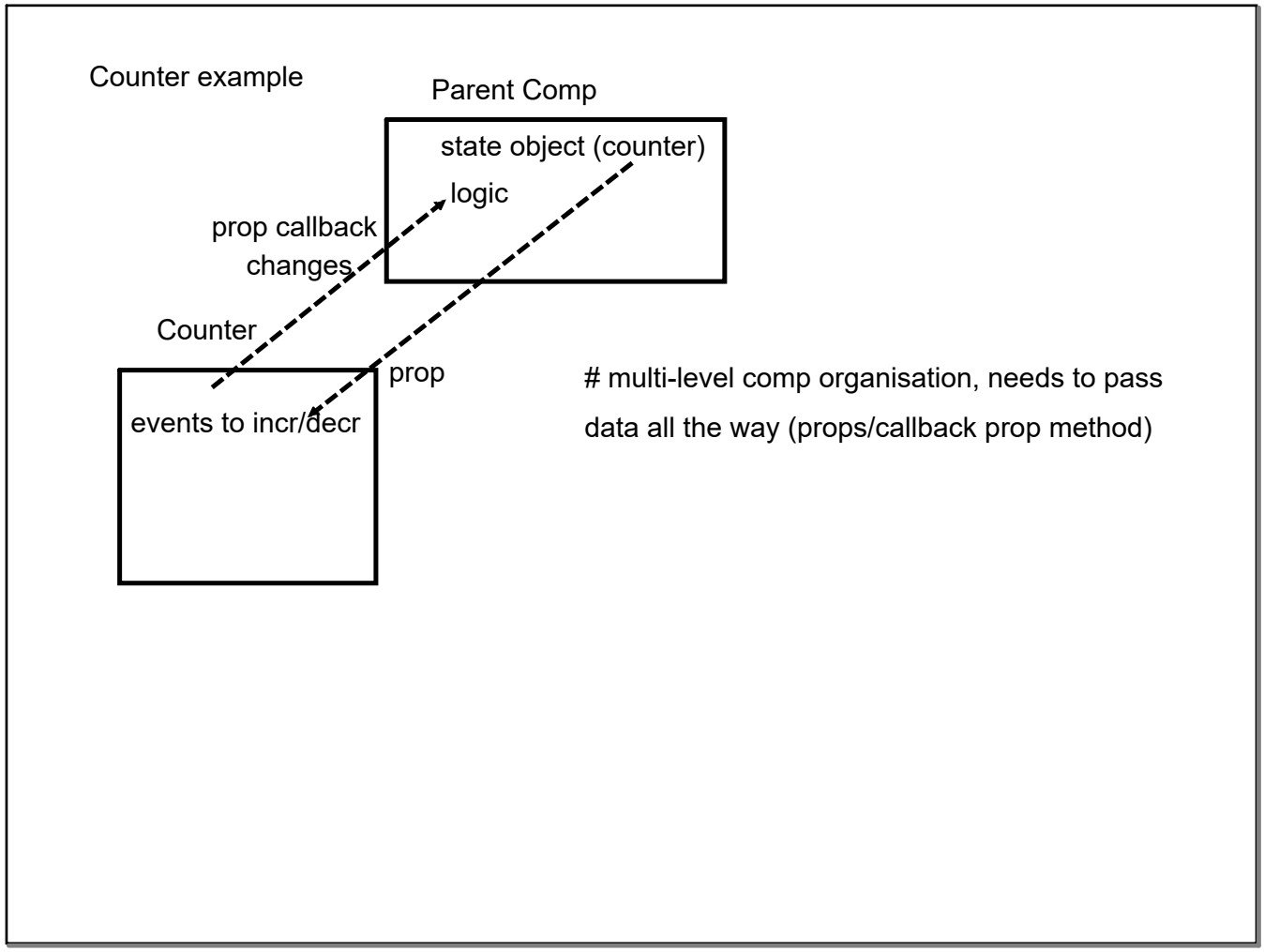
we need to push new url to address bar



Control flows from higher level component down to child comp, forcing changes to happen reactively (modular, fast (diffing identification is efficient))

=>Parent comp can send data to child comp(props)

=> child comp can accept callback methods(props) to communicate back with parent comp


Counter example
Parent Comp
state object (counter)
logic
prop callback
changes
Counter
prop
events to incr/decr
multi-level comp organisation, needs to pass
data all the way (props/callback prop method)

Single event triggers off sequence of events

==> List of email

Event : click on any email

==>sequence of event

replace inbox view (list) with email view (single email)

fetch email data from server

mark that email as read

reduce the counter of unread mail

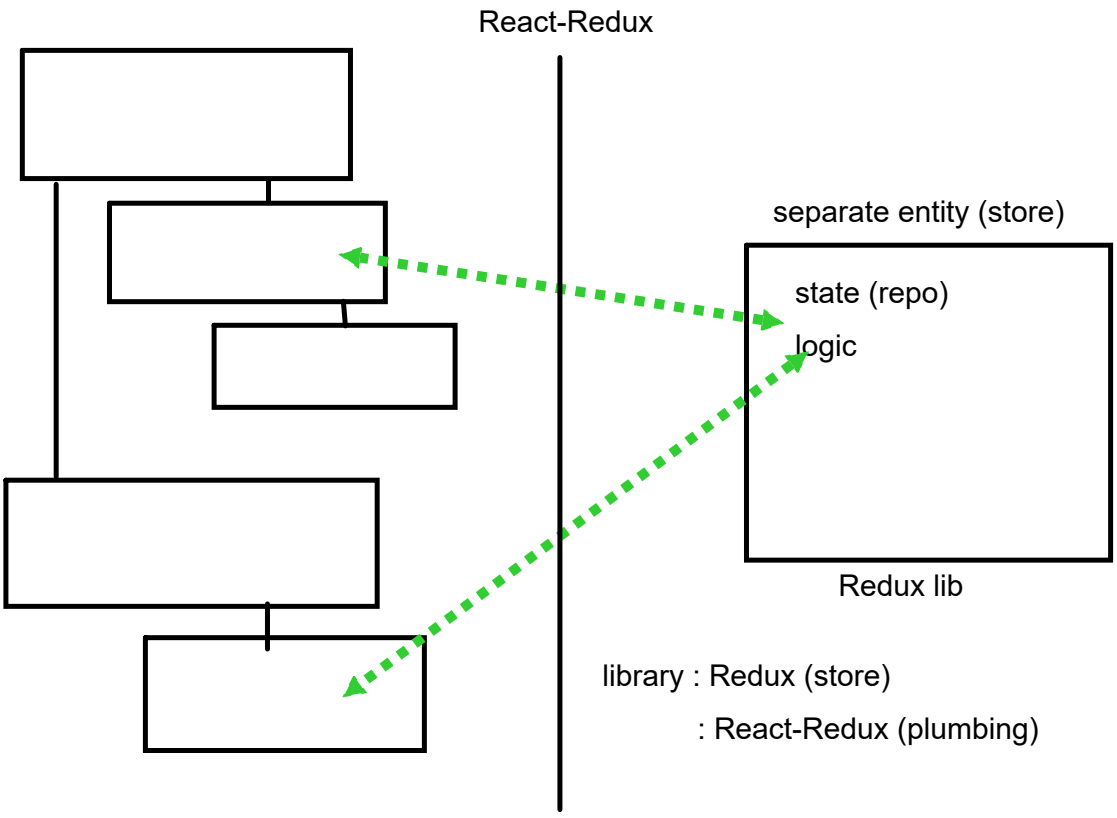
change in url

update all this on server

Flux Design pattern:

Present : all repo/logic is managed at top level by component

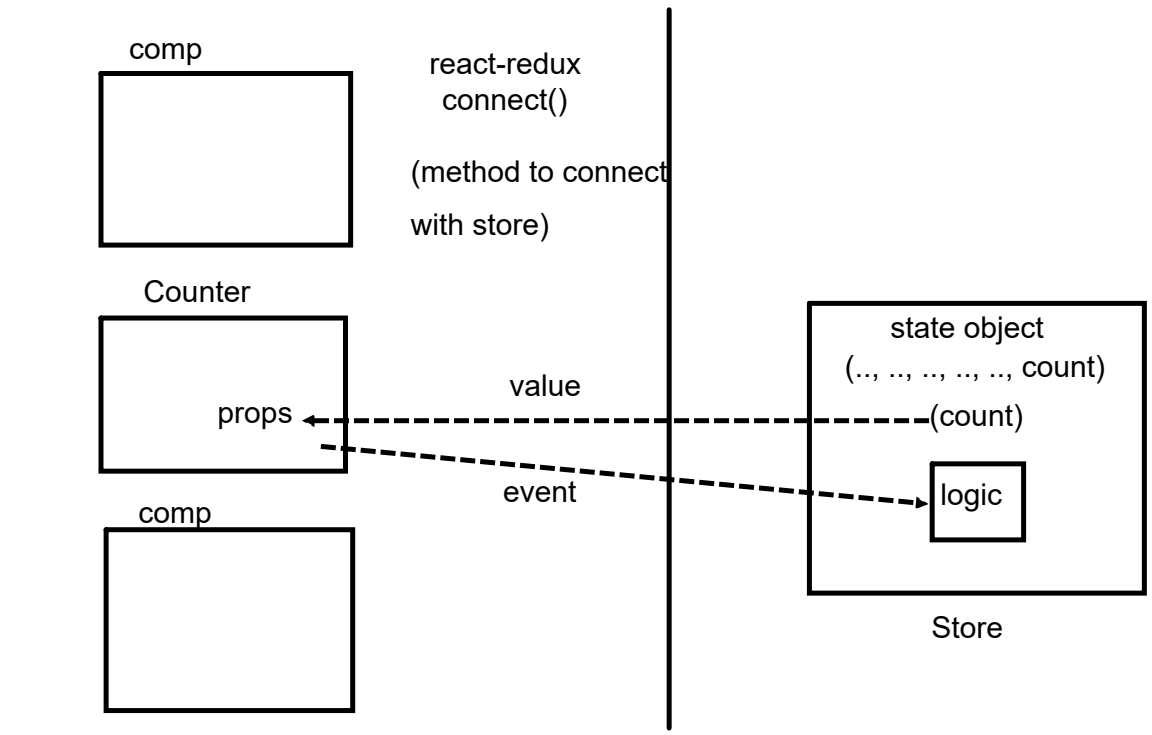
Requirement : separate entity responsible for maintaining state (repo) and the logic to go with it..



Redux : JS store

installing redux lib

> npm install redux react-redux



1.. an independent method that decide/define what part of state object (store) we need to use (mapping State to props)

2. connect (react-redux) helps to connect Component with store

connection will be made using logic of mapping function

> npm add redux react-redux --prod