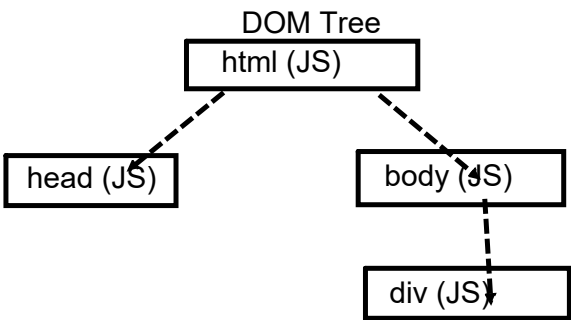


React JS : V of MVC

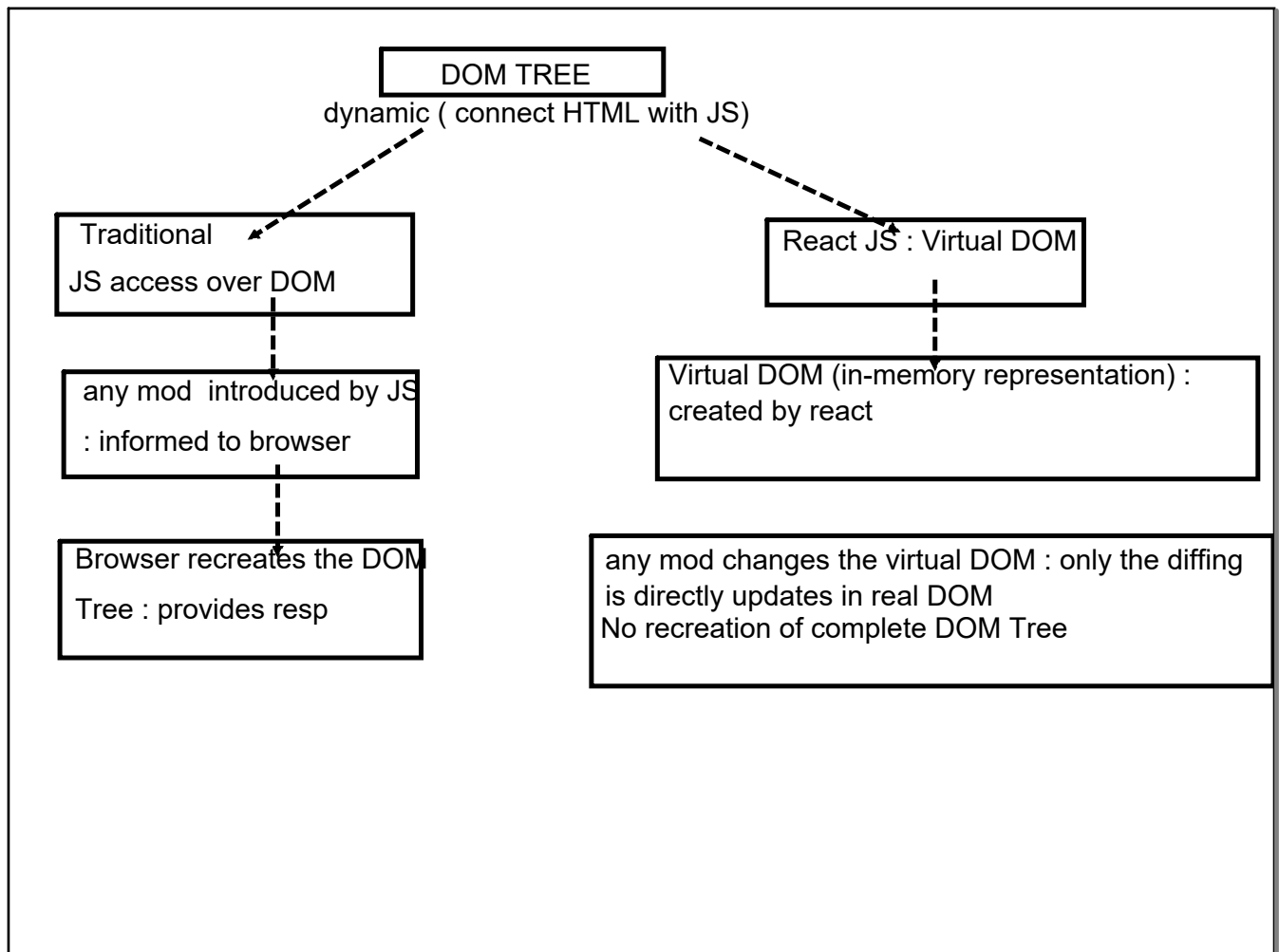
Angular : framework : MVC/W

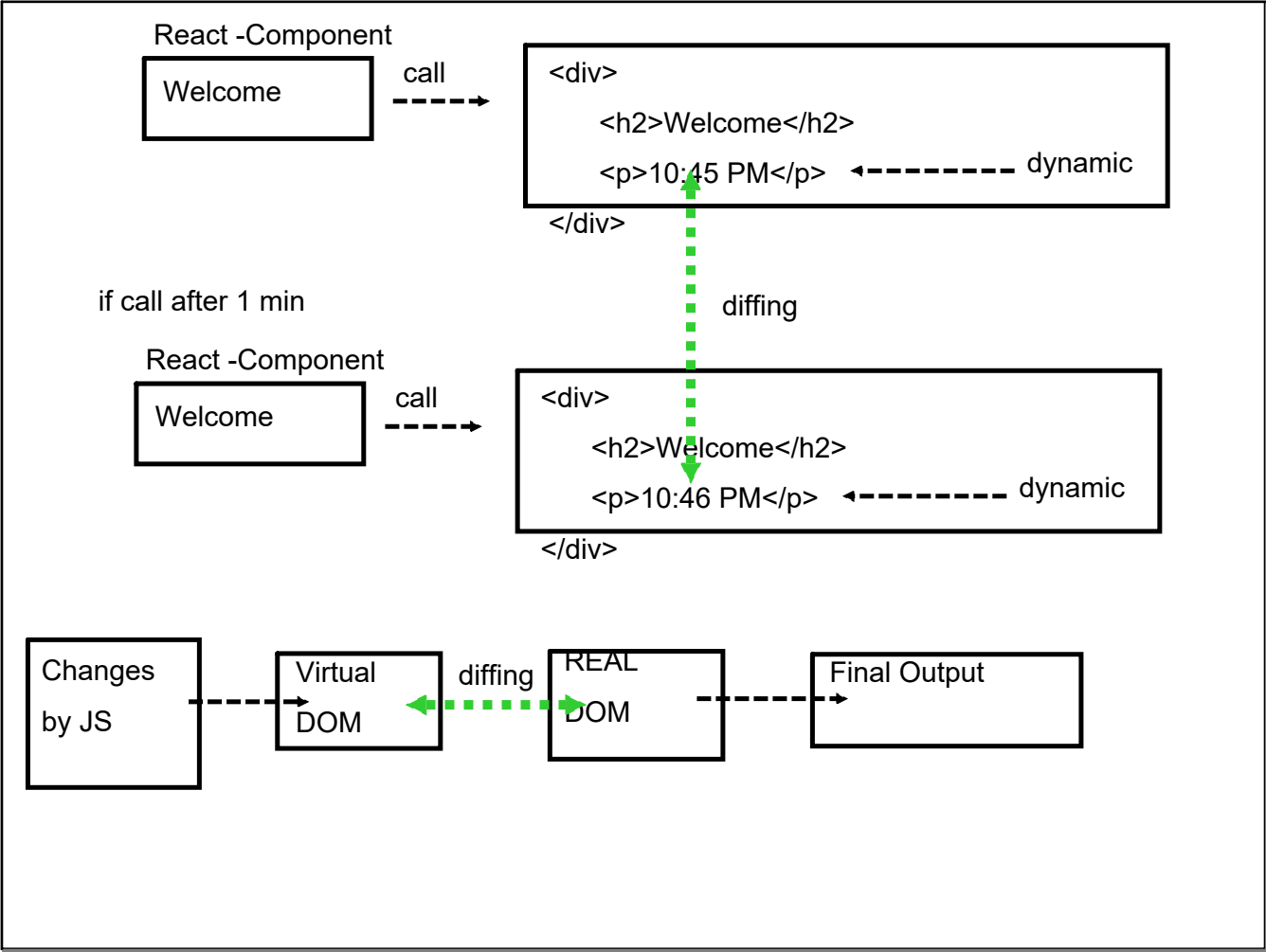


DOM : Document Object Model

Every HTML Component is converted into JS Object

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





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:

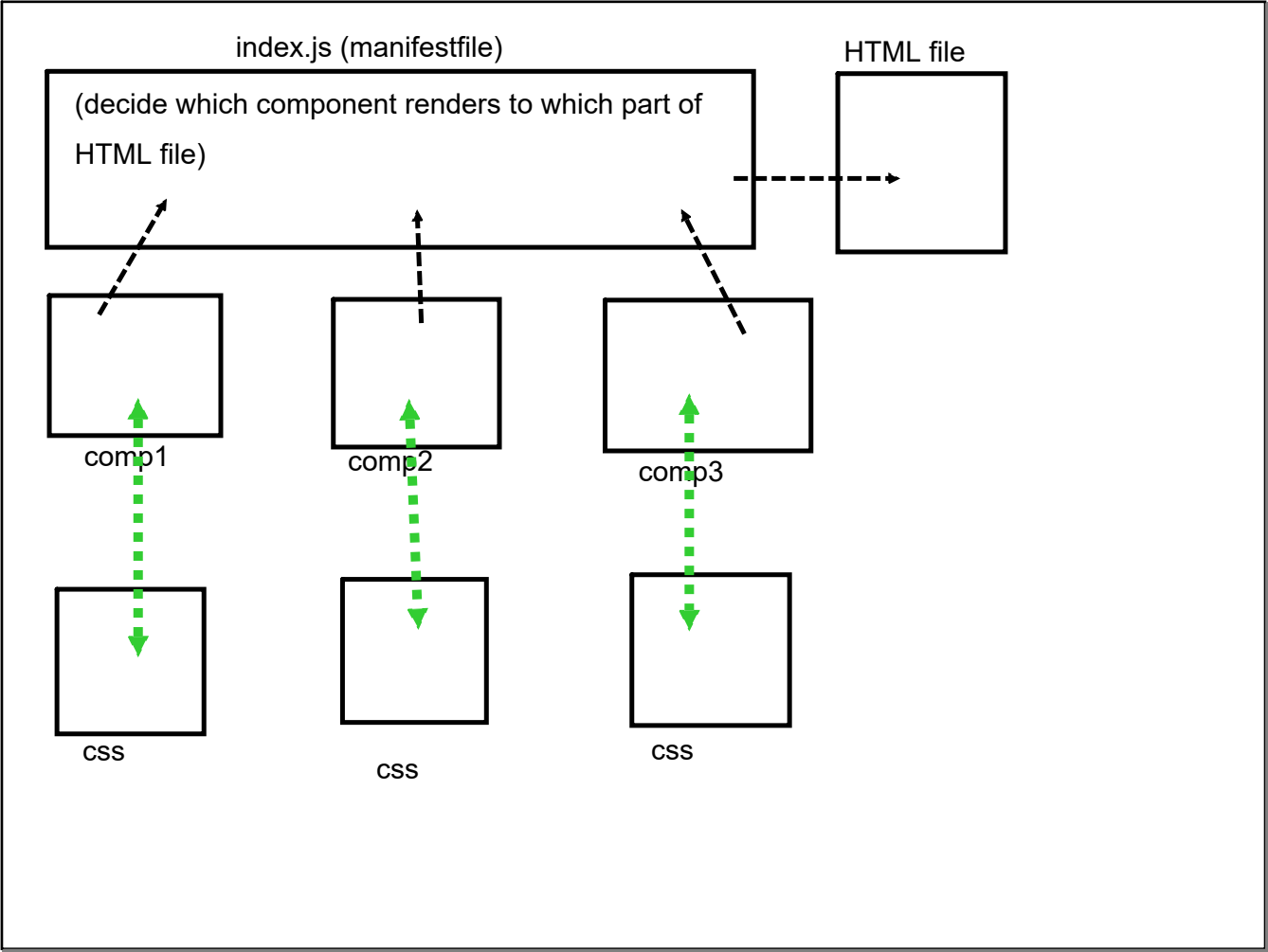
```
>create-react-app <project-name>
```



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)



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

Launching React app : `npm start`



Creating React Component:

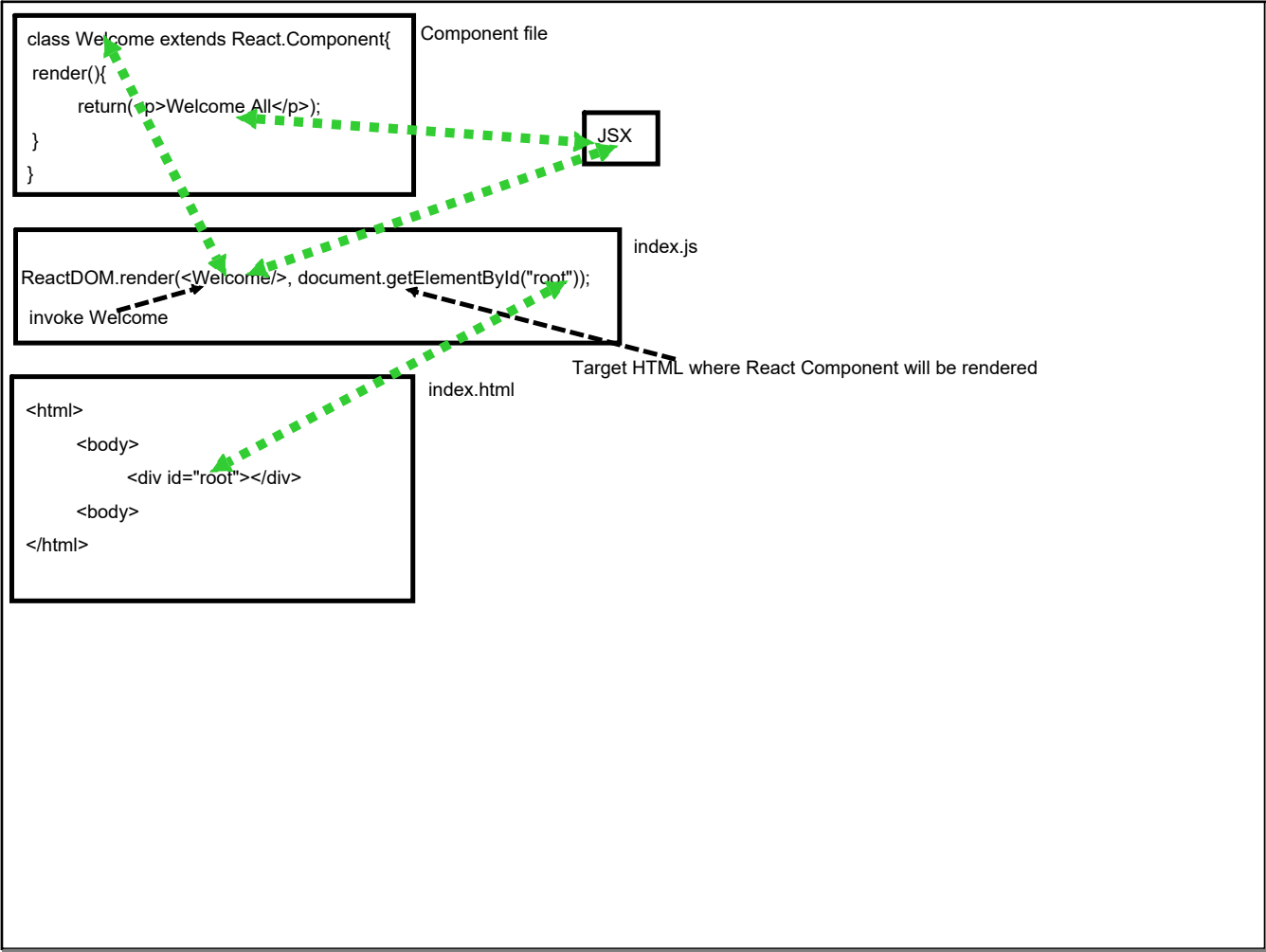
# React comp : JS class

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

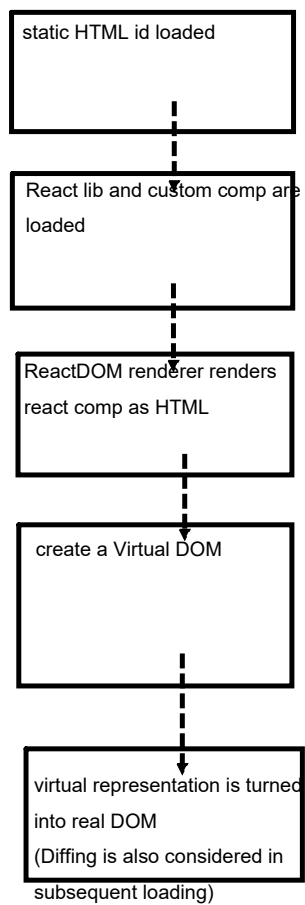
```
class Welcome extends React.Component{  
  // method needs to override  
  render(){  
    // must return the dynamic UI to render  
    return (<JSX Code>)  
  }  
}
```

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

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



## React Application Flow



JSX ( Javascript XML ) : ( XML in background maintaining relationship between 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') );
```

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

Generated HTML

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

Comment Box

Entry form for adding new comment

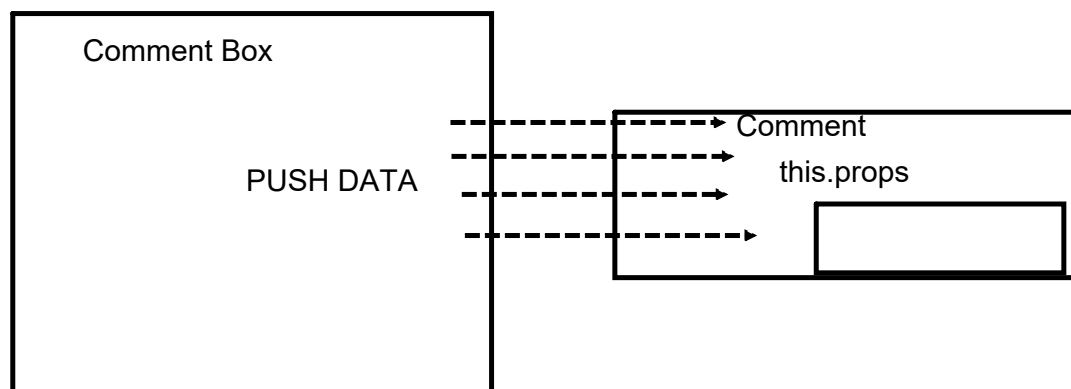
Show/Hide

Author Name : \_\_\_\_\_

delete X

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)

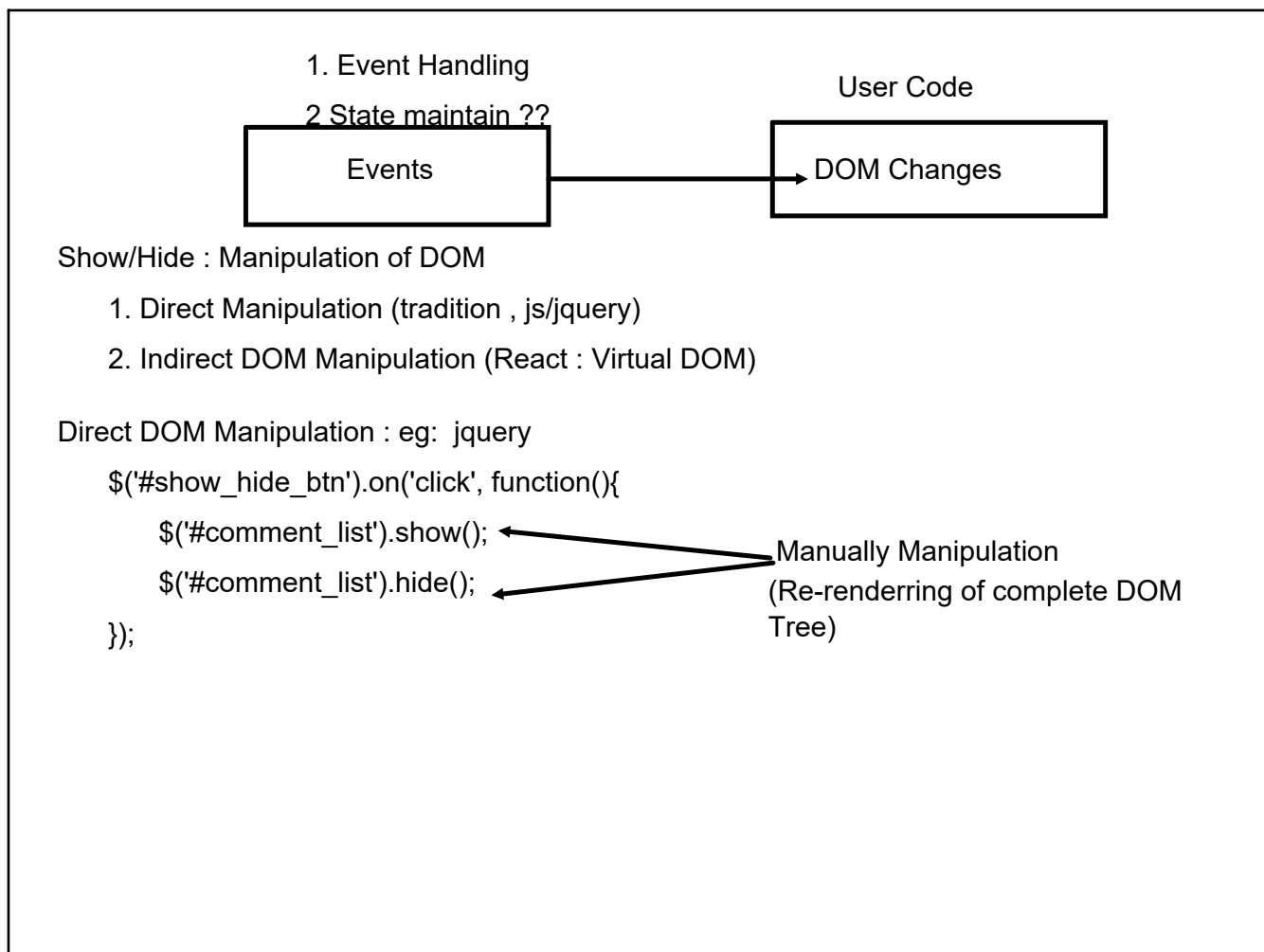


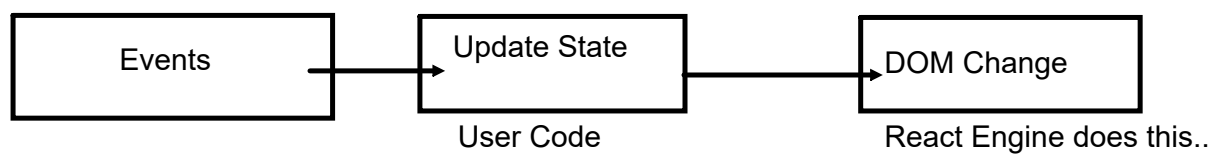
Type of prop values:

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

React JS by default provides one way data flow

Parent to child





=> 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)

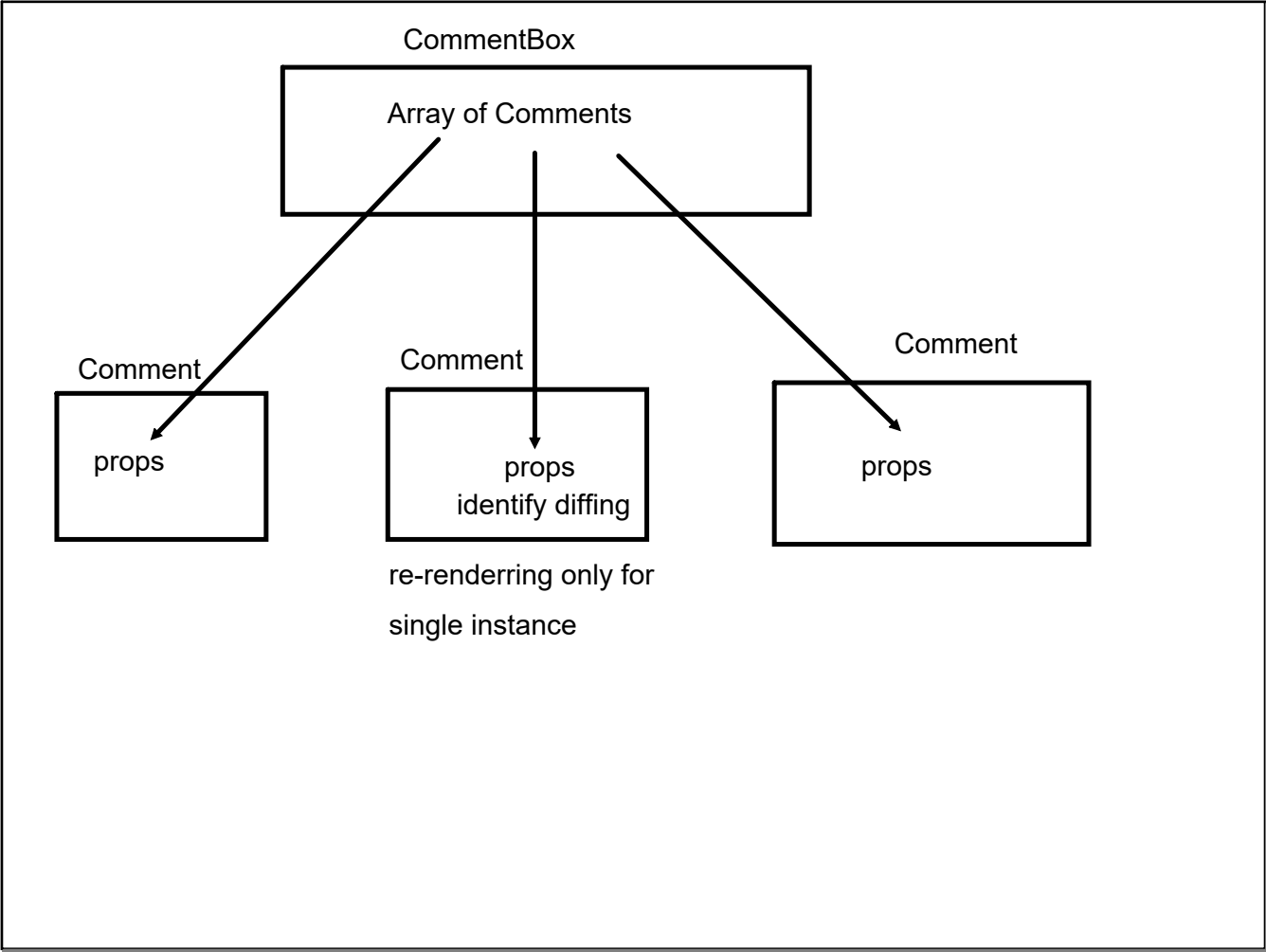
==> we need to add custom fld

==> constructor : set the initial state

Trigger of render takes place in following cases...

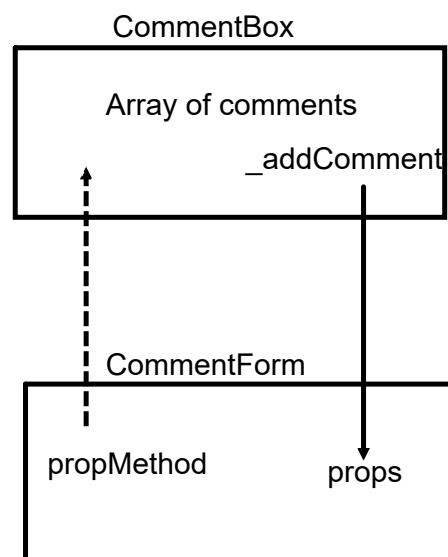
1. Changes in state object
2. any modification in props values
3. any modification in instance members

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



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

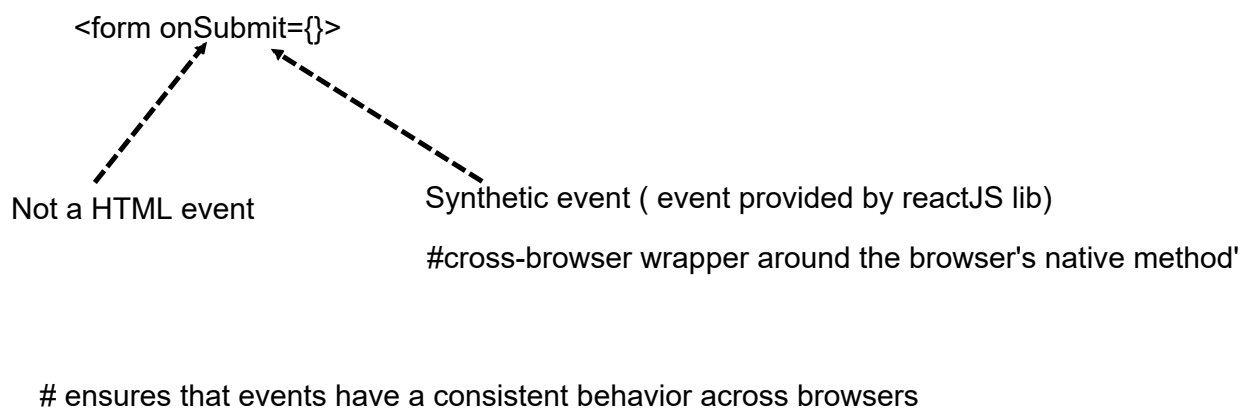
==> Need to have the values of form fields available as instance member of Component class ( so that we can decide upon what to do with form fields)

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

`<input placeholder="Author:"` a new instance member would be created  
`ref={(input) => this._author = input}/>`

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





# 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 rendering (once)

`componentWillUnmount()`; just before component is removed

# polling : periodically check for updates : continuous call of `fetchComment()` at regular interval

# Need to initiate a polling process : `componentDidMount()`

Lots of component

there will be continuous 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

