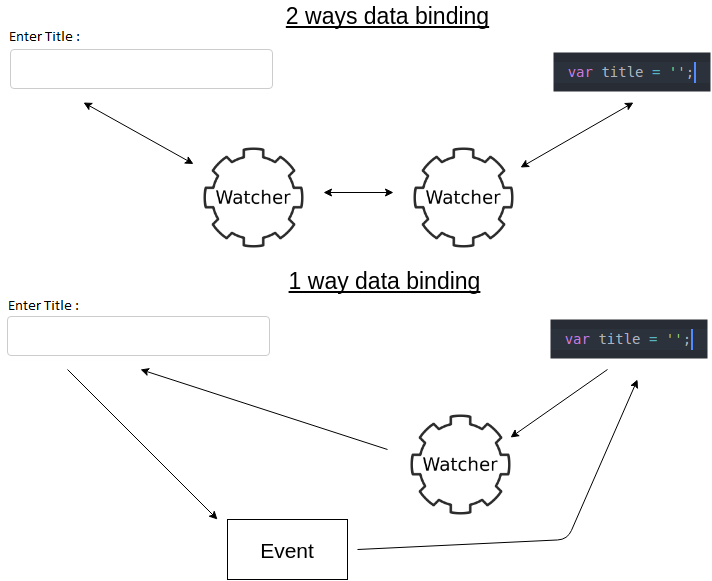
**React**

React Js is an open source JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies.

React can be used as a base in the development of single-page or mobile applications.

React uses One-way data flow or binding:

* This is called Unidirectional data flow.
* Unidirectional data flow means that React is more performant than Angular
* React doesn’t have a mechanism to allow the HTML to change the component. The HTML can only raise events that the component responds to.



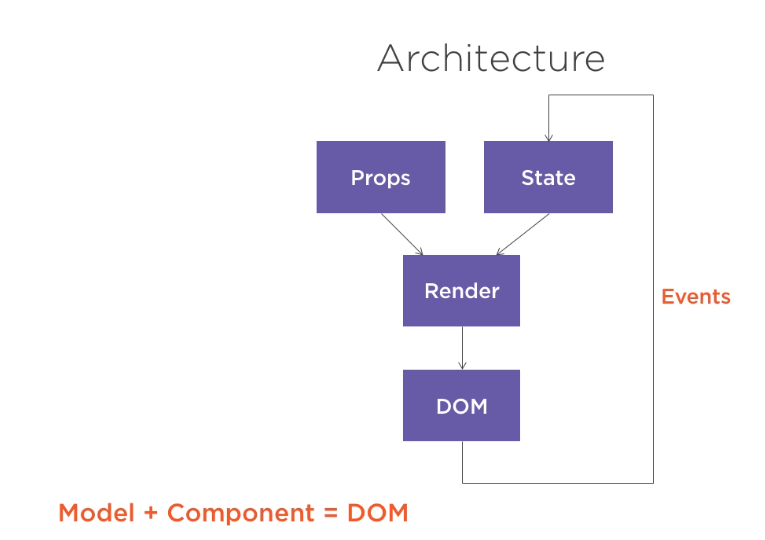
Create React App

<https://github.com/facebook/create-react-app>

Steps:

1. npm install -g create-react-app
2. create-react-app <Name of the Application> // Will create Skelton of the application
3. Change directory to <Name of the Application>

* To start : npm start
* To run build : npm run build
* To Test : npm test



VDOM?

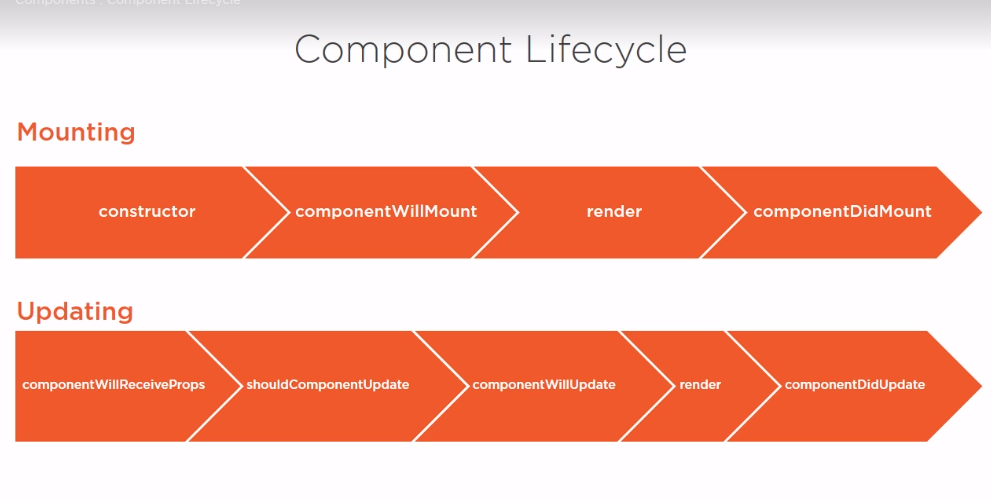
The Virtual DOM (VDOM) is a programming concept where an ideal or virtual representation of a UI is kept in memory and synced with the real DOM by a library such as ReactDOM. This is process caller **reconciliation**.

**React Components**

Component are fundamental unit of a react application. Each component corresponds an element in the DOM. Component is response for rendering the content of that element and handling any events with in it.

JSX is a markup language looks like Html syntax but can be specify inside the javascript code. There Transpilers to compile JSX into javascript.

Component Lifecycle



We can pass data from parent component to child component using “props” and can pass/call data to the parent component using “events”

Other than “**props**” we can hold the data using “**state**”. **State is local and mutable data that can be created and modify within the component**.

Example :

**class ClickCounter extends React.Component {**

**constructor(props)**

**{**

**super(props);**

**this.state = {clicks : 0};**

**}**

**render() {**

**return (**

**<div>**

**<h1 onClick={() => {this.setState({clicks: this.state.clicks + 1});}}>Clicked {this.state.clicks} times</h1>**

**</div>**

**);**

**}**

**}**

**ReactDOM.render(**

**<ClickCounter />, document.getElementById('app')**

**);**

**Validating Props :**

React component can validate the props that passes to it using **PropType (prop-type)**.

Npm install prop-type

Import **PropType from ‘prop-type’**

We can have all the prop type here <https://www.npmjs.com/package/prop-types>

Example :

import React from 'react';

import PropType from 'prop-types'

class AuthorQuiz extends React.Component {

  render() {

    return (

      <div>

        Author Quiz  {this.props.name} - {this.props.age}

      </div>

    )

  }

}

AuthorQuiz.propTypes = {

  name: PropType.string,

  age: PropType.number

};

export default AuthorQuiz;

**Testing Reacjs :**

Create-react-app default uses the test package “Jest”

Enzyme module helps to write unit test effectively.

Npm install –save-dev enzyme-adapter-react-16

JSX:

* Allows xml like markup in side javascript
* JSX transformer transform JSX syntax to javascript function call.
* Babel or Type script compiler is tranculds JSX into javascript
* Example 1:
  + **JSX : <Sum a={4} b={3} />**
  + **Javascript : React.createElement(Sum, {a:4, b:3}, null);**
* Example 2:
  + **JSX : <h1> <Sum a={4} b={3} /> </h1>**
  + **Javascript : React.createElement(‘h1’, null, Reacr.createElement(Sum, {a:4, b:3}, null));**
* Spread Syntax : JSX supports spread syntax
  + Example :

Const props = {a:4, b:2};

Const element = <Sum {…props}

* When we pass a html content to a com[onent by default react will display that content in Html syntax not html element. If we want it in html element, we can use **dangerouslySetInnerHTML** with \_\_html syntax
  + Example :
    - Return <p dangerouslySetInnerHTML = {{\_\_html: props.htmltext}}
* JSX component can be nested with one another and using **props.children** we can get the child components of a component.
* If we already render the child components react automatically eliminates duplicate component. We can conditionally display child components
* Example

**Function ConditionalDisplay(props)**

**{**

**Return(**

**<div> {props.isVisible ? props.children : null} </div>**

**)**

**}**

**Function main()**

**{**

**Return {**

**< ConditionalDisplay isVisible ={true}>**

**<div>Test</div>**

**< /ConditionalDisplay>**

**}**

**}**

* Here if isVisible = true then only <div>Test</div> will be displayed.

Some Javascript help :

* Object.Map : Used to loop through a array and returns a new array

Example:

Var array1 = [1,4,9,16];

Const map1 = array1.map(x => x\*2); //2,8,18,32

* **Object.reduce :** Executes a reduce function on each element of the array, resulting in a single value

Example :

Const array1 = [1,2,3,4];

Const recederfun = (accumulator, currentValue) => accumulator + currentValue

Console.log(array1.reduce(recederfun)); // Answer is 10

Console.log(array1.reduce(recederfun,5)); // Answer 15 1+2+3+4+5

Reducer function takes four arguments

* + Accumulator
  + CurrentValue
  + CurrentIndex
  + SourceArray

- **Object.slice** - Returns the selected elements in an array as a new array from begin to end

Var animal = [‘ant’,’bison’,’camel’, ‘duck’, ‘elephant’];

Console.log(animal.slice(2)) ; //’camel’, ‘duck’, ‘elephant’

Console.log(animal.slice(2,4)) // ’camel’, ‘duck’

**Events:**

**class Head extends React.Component {**

**render() {**

**/\*const clickHandler = (synthEvent) => {**

**console.log(synthEvent)**

**};\*/** // Here clickHandler is a method that accepts a input parameter also

**//** console.log is a method that accept one parameter so we can write this as below

**const clickHandler = console.log;**

**return (**

**<div>**

**<button onClick={clickHandler}> Make an event </button>**

**</div>**

**)};**

**}**