React JS

By Akash Kale

WHAT IS REACT?

- A JavaScript Library For Building User Interfaces
- Renders your UI and responds to events.
- It also uses the concept called Virtual DOM, creates an in-memory data structure cache, enumerates the resulting differences, and then updates the browser's displayed DOM efficiently.
- One of the unique features of React.js is not only it can perform on the client side, but it can also be rendered on the server side, and they can work together interoperably.

WHAT IS REACT?

Angular has

- modules
- controllers
- directives
- scopes
- templating
- linking functions
- filters
- dependency injection

WHAT IS REACT?

~~Angular~~ React has **JUST COMPONENT**

- ~~modules~~
- ~~controllers~~
- ~~directives~~
- ~~scopes~~
- ~~templating~~
- ~~linking functions~~
- ~~filters~~
- ~~dependency injection~~

React isn’t a complete framework.

React is just the V in MVC.

Pre-requisites for React JS

- HTML/CSS
- JavaScript
- Node JS
- Babel Transpiler
- TypeScript
- Web Technologies (Servlet/JSP, PHP, ASP.net – Optional)

Software's required

- Node.js 6.9.0 or later
- NPM 3.0 or later (comes with Node.js).
- Editor Visual Studio Code or Brackets or Sublime.
- Android Studio – > android mobile (Native React)
- Xcode –> ios mobile (Native React)
- Rest API

Why node.js ?

“Node's goal is to provide an easy way to build scalable Network programs”

What is NodeJS ?

- NodeJS is an open source , cross platform runtime environment for server side and networking application.
- It is written in JavaScript and can run on Linux , Mac ,Windows , FreeBSD.
- It provided an event driven architecture and a non blocking I/O that optimize and scalability. These technology uses for real time application.
- It used Google JavaScript V8 Engine to Execute Code.
- It is used by Groupon, SAP , LinkedIn ,Microsoft,Yahoo ,Walmart ,Paypal.

What is unique about Node.js?

- JavaScript used in client-side but node.js puts the JavaScript on server-side thus making communication between client and server will happen in same language
- Servers are normally thread based but Node.JS is "Event" based. Node.JS serves each request in a Event loop that can able to handle simultaneous requests.
- Node.JS programs are executed by V8 JavaScript engine the same used by Google chrome browser.

Node.js is not……

- Another Web framework
- For beginner
- Multi-thread

In Simple Node.js

Node.js = Runtime Environment + JavaScript library.

Why node.js ?

- Non Blocking I/O
- V8 JavaScript Engine
- Single Thread with Event Loop
- 40,025 modules
- Windows, Linux, Mac
- 1 Language for Frontend and Backend
- Active community

Why JavaScript ?!!!

- Friendly Callbacks
- No I/o Primitives
- One language to RULE them all

JavaScript is well known for client-side scripts running inside the browser

node.js is JavaScript running on the server-side

SSJS -> Server-Side JavaScript

Use a language you know

Use the same language for client side and server side

ES6

- JavaScript is ES6 or ES 2015 as well as ES7 or ES2016, ES means ECMAScript which is a standard for scripting language ECMA means European Computer Manufacturer's Association.
- The ES6 or ES6 script can’t understand by browser.
- So it required transpiler, which help to convert ES6 or ES7 to ES5(JavaScript).
  - TypeScript transpiler
  - Babel transpiler
  - Traceur transpiler

TypeScript

- *TypeScript* is a free and open source programming language developed and maintained by Microsoft.
- It is a strict superset of JavaScript, and adds optional static typing and class-based object-oriented programming to the language.
- It is designed for development of large applications and transpiler to JavaScript. As TypeScript is a superset of JavaScript, any existing JavaScript programs are also valid TypeScript programs.

Features of the TypeScript

- Data Types Supported
- ES6 function
- Classes
- Inheritance
- Interface
- Modules
- Source File Dependencies

Variable declaration

- In ES6, we can use var, let and const keyword to declare the variable.
- var keyword is used to declare the function scoped variable.
- let keyword is used to declare the block scoped variable.
- const keyword is used to declare the constant value.

Data Types

- Any
- Primitive
  - number
  - boolean
  - string
  - void
  - null
  - undefined – Same as JS
- Array
- Enum

TypeScript functions

- Normal function
- Function with parameter
- Function with parameter variable with datatype
- Function with return types.
- Function with required parameter
- Function with default parameters
- Function with optional parameters

Continue...

- Rest parameter
- Spread parameter
- Arrow function (lambda expression)

ES6 String

- let msg1="Simple Msg"; //double quote
- let msg2='Simple Msg'; // single quote
- let msg3=`Simple Msg`;//backticks quote
- let name ="Raj Deep";
- let msg4 = `The name is
  ${name} `; //string template

Different way for loop

- let abc=[“a”,”b”,”c”,”d”]
- First way

```
for(let i=0;i<abc.length;i++){
console.log(abc[i])
}
```

- Second way

```
for(let index in abc) {
console.log(abc[index])
}
```

Continue...

- Third way

```
for(let val of abc) {
console.log(val)
}
```

- Fourth way

```
abc.forEach(function(value){
console.lot(value)
})
```

using arrow function

- Fifth way

```
abc.forEach((value)=>console.log(value))
```

Classes

- Contains set of properties and behaviour
- Instance methods/members
- Static methods/members
- Can implement interfaces
- Inheritance
- Single constructor
- Default/Optional parameter
- ES6 class syntax

Inheritances

- Using classes
- Using interface

Source File Dependencies

- Can be done using reference keyword
- Must be the first statement of file
- Paths are relative to the current file
- Can also be done using tsconfig file

Example Demo

```
///<reference path=“./a.ts”>
///<reference path=“./b.ts”>
abc();
xyz();
```

Modules

- Modules can be defined using module keyword
- A module can contains sub-modules, class, enum or interfaces. But can’t directly contains functions.
- Modules can be nested(sub-modules).
- Classes and Interfaces can be exposed using export keyword.

Types of module

- Internal module
- External module

Sample Example internal module

- **a.ts**

```
module A{
    export function abc(){
    console.log("Abc function
from a.ts")
    }
}
```

- **b.ts**

```
module B {
    export function abc() {
    console.log("Abc function
from b.ts")
    }
}
```

- **main.ts**

```
///<reference
path="./a.ts"/>
///<reference
path="./b.ts"/>
A.abc();
B.abc();
```

Sample Example external module

- **a.ts**

```
    export function abc(){
console.log("Abc function from
  a.ts")
      }
```

- **b.ts**

```
  export function abc() {
console.log("Abc function from
  b.ts")
      }
  }
```

- **main.ts**

```
import {abc} from "./a";
import {xyz} from "./b";
abc();
xyz();
```

React JS

- React is developed by Facebook and React has good community support for developers who uses React to build the web pages.
- Many of the Single page applications nowadays use Component based approach like GitHub, Facebook, Udemy, Angular official webpage, React official webpage, Gmail and etc.

SPA Vs MPA

- In Single Page Application, only a portion of the content is updated. Thus, there is no need to re-download the same elements.
- High speed.
- Good to develop mobile application very fast

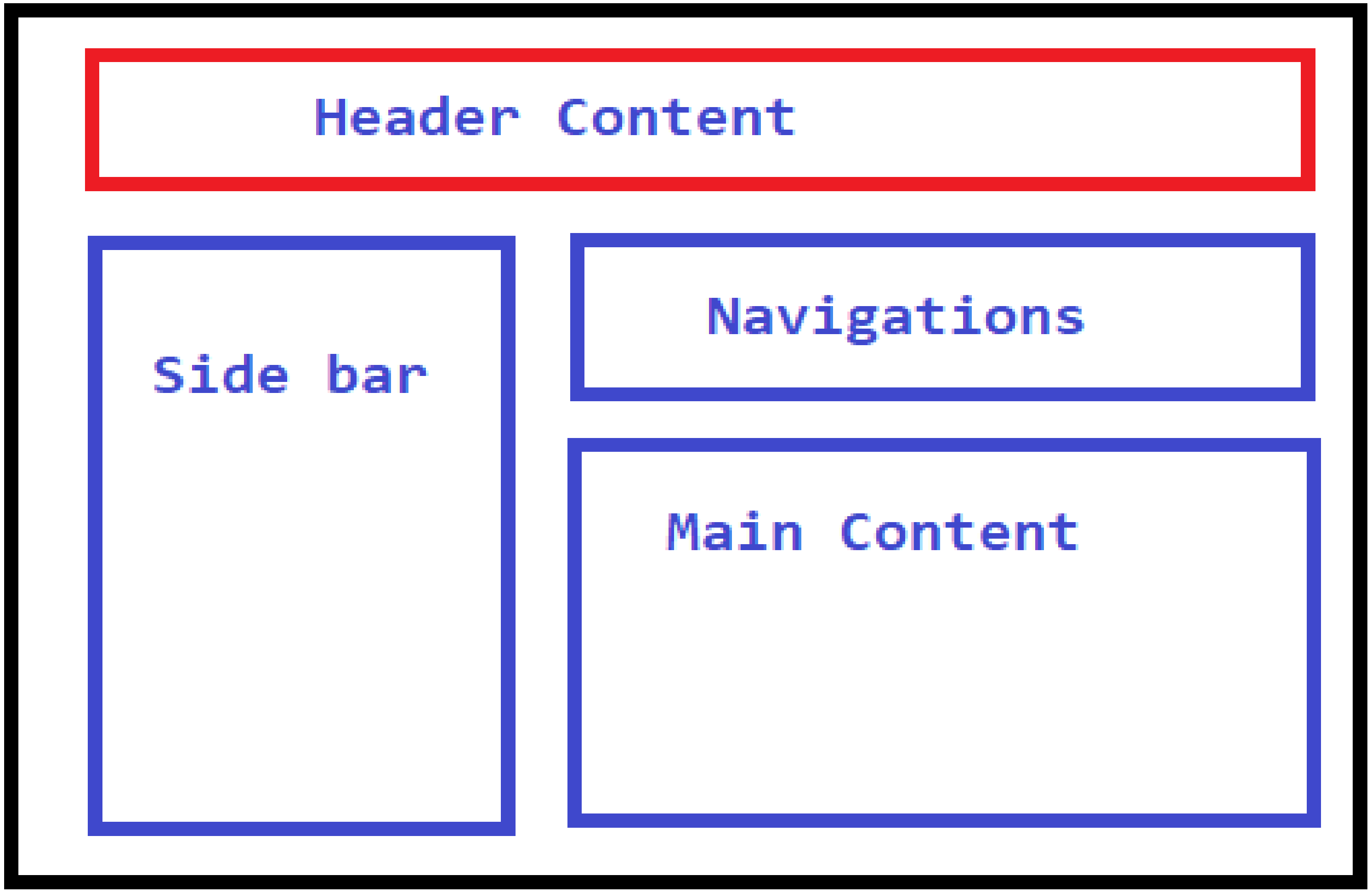
- In Multiple Page Application, Each page sends a request to the server and completely **updates all the data**. Even if these data are small.
- Takes more time to download the page.
- Take more time to developer mobile application

React JS

- But just a library for creating user interfaces.
- Renders your UI and responds to events.
- a.k.a. The “V” in **M V C**
- User Interfaces are basically what the user sees on the web page, React is all about using components to build these User Interfaces.

React Component

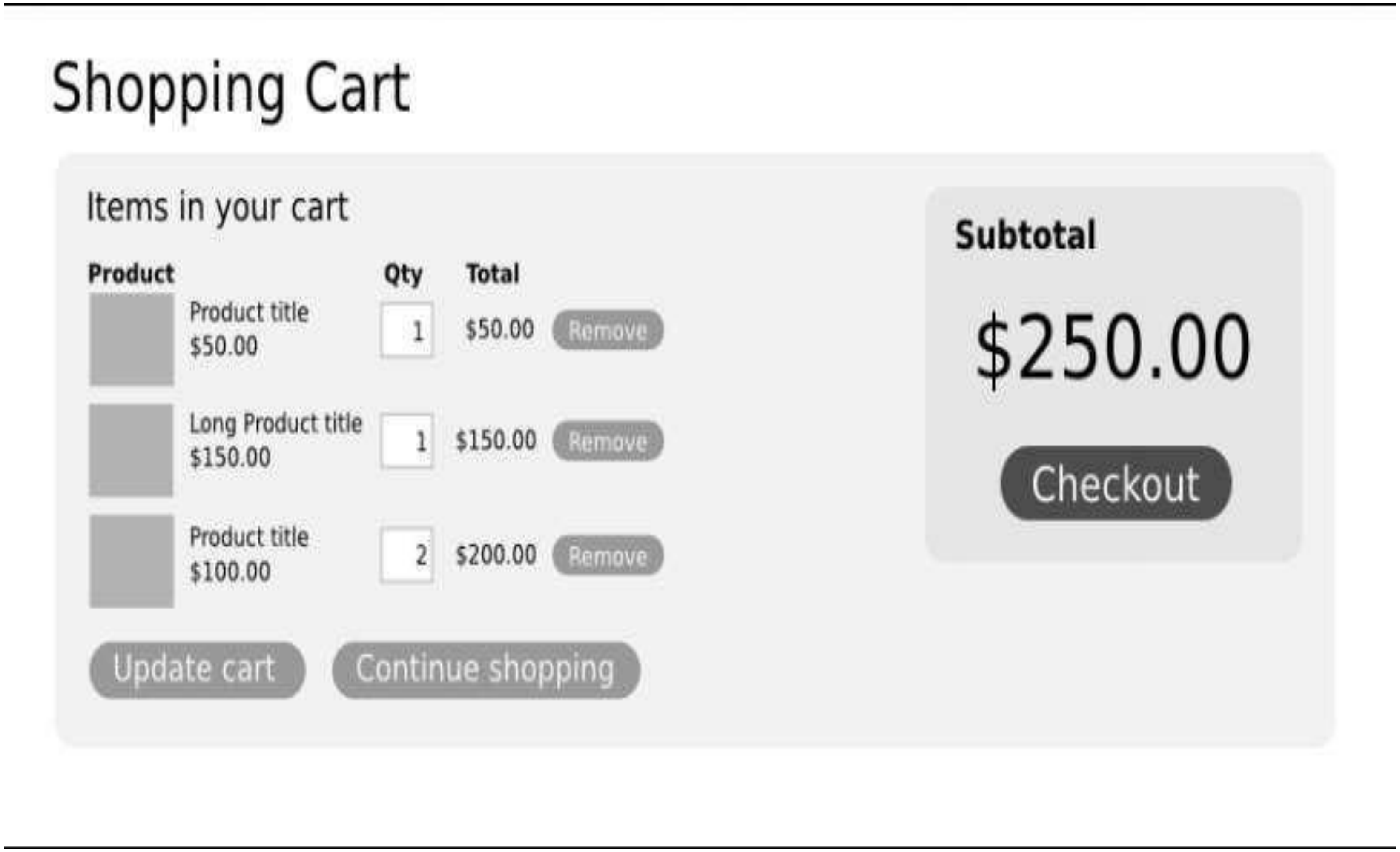
- Components: These are part of the webpage which user sees, a web page is made up of multiple components and every component can be independent from other component. If you think of a web page it can be split it into multiple sections like this


Header Content
Side bar
Navigations
Main Content

COMPONENT

- Components let you split the UI into independent, reusable pieces, and think about
- each piece in isolation.

- Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called "props") and return React elements describing what should appear on the screen.



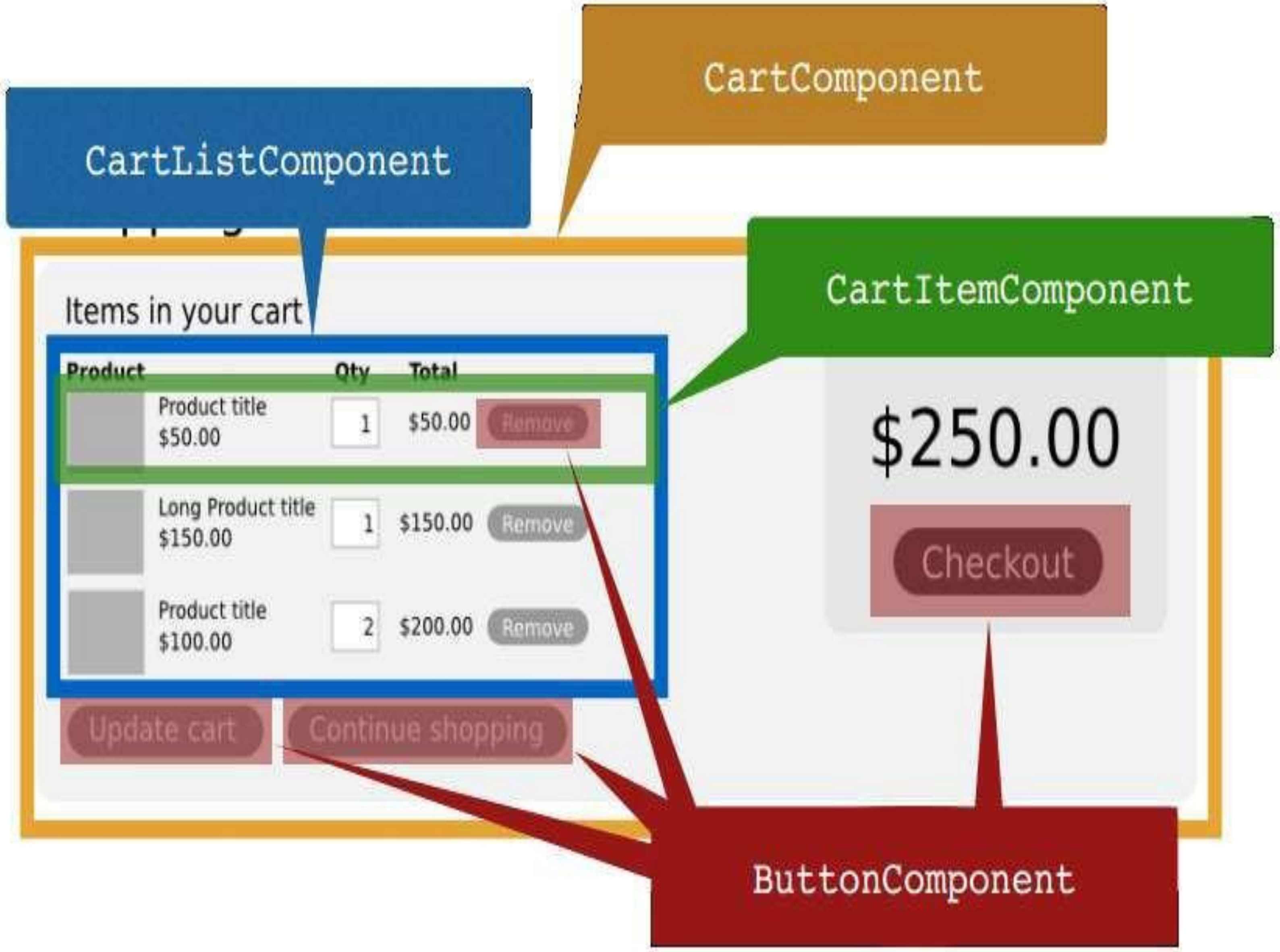
Shopping Cart

Items in your cart

Product		Qty	Total	
	Product title $50.00	1	$50.00	Remove
	Long Product title $150.00	1	$150.00	Remove
	Product title $100.00	2	$200.00	Remove

Subtotal

$250.00


CartComponent
CartListComponent
CartItemComponent
Items in your cart
Product
Qty
Total
Product title
$50.00
1
$50.00
Remove
Long Product title
$150.00
1
$150.00
Remove
Product title
$100.00
2
$200.00
Remove
$250.00
Checkout
Update cart
Continue shopping
ButtonComponent

React JS with ES5

- Plain Javascript code
- To render the content in plain Javascript code you must create the element using React.createElement() which takes 3 parameters minimum
- 1st parameter is the element name
- 2nd parameter is the JavaScript object to represent stylings to the element
- 3rd parameter could be child element or text
- React.createElement("h1", null, "Hello World");
- ReactDOM.render(eleRef,document.getElementById("root"))

JSX

- JSX means JavaScript Extension.
- JSX is a XML like syntax extension of ECMAScript without any defined semantics.
- It looks like a XML.
- JSX is code looks like HTML but is actually a mix of JavaScript and HTML code.

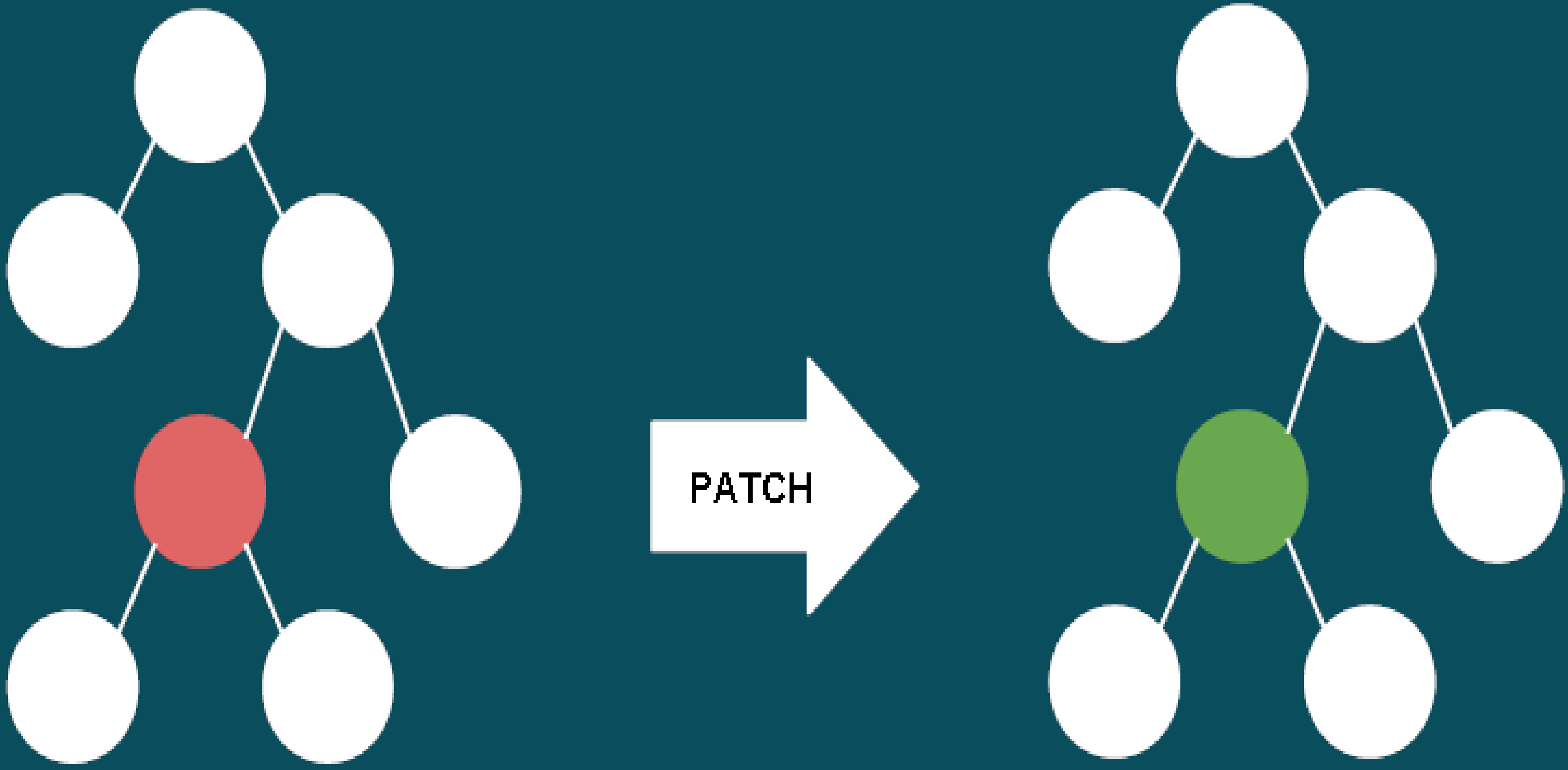
Continue...

- JSX hence we need a compiler to convert the JSX to Javascript code one such compiler is Babel.js which you need to include in the script i.e.,
- <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>

Virtual DOM

- Makes re-rendering on every change fast.
- On every update…
- React builds a new virtual DOM subtree
- …diffs it with the old one
- …computes the minimal set of DOM mutations and puts them in a queue
- …and batch executes all updates.
- Virtual DOM is simple and fast.

VIRTUAL DOM   REAL DOM



virtial dom tree

real dom tree

COMPONENT - PROPS

- Props is what you pass into the Component via attributes.
- Props is the only way to input data. (Or you can use Redux).
- Props are immutable.
- Container component will define data that can be changed
- Child Component will received data from parent component via props.

COMPONENT - STATE

- Private data of component
- When change -> Re-render Component
- Can’t read from outside Component

React JS with ES6

ES5

```
function App() {
return
  <h1>ReactJS</h1>
}
```

ES6

```
class App extends
  React.Component {
  render() {
return
  <h1>ReactJS</h1>
  }
}
```

React JS project with node js

- *create-react-app application-name*
- You can navigate inside your application and hit *npm start* command which deploys your application in the development server and automatically opens the browser, it runs in 3000 port number.
- like
  - cd application-name
  - npm start

State and props

- State and Props are inbuilt properties of a Component which you can access using this keyword
- Props is an object which must not be changed and it is passed from another component, State is another object of a Component which can be changed.

Continue...

- Set the properties in state
- this.state(“key1”,”value1”,”key2”,”value2”);
- Change the properties from a state
- this.setState({“ke1”,”newValue1”,”key2”,”newValue2”})

Handling events

- Handling events with React elements is very similar to handling events on DOM elements. There are some syntactic differences:
  - React events are named using camelCase, rather than lowercase.
  - With JSX you pass a function as the event handler, rather than a string.

JavaScript event Vs React JS

- <button onclick=“dis()"> Activate Lasers </button>
- <button onClick={this.dis}> Activate Lasers </button>

Continue...

- Another difference is that you cannot return false to prevent default behavior in React. You must call preventDefault explicitly. For example, with plain HTML, to prevent the default link behavior of opening a new page, you can write:

  -
    ```
    function handleClick(e)
    { e.preventDefault();
    console.log('The link was clicked.');
    }
    ```

React Form handling

- In HTML the form internally keep some state with name attribute.
- However in React mutable state is kept in the state property of Components, and only updated with setState().
- Function to handle the change

```
  handleChange(event) {
     let inputName  = event.target.name;
     let inputValue  = event.target.value;
    this.setState({[inputName]:inputValue})
}
```

React JS Life cycle

- Components go through certain life cycle method which are called automatically when something happens to the component which are also called as hook, these methods you can override to run code at particular times in the process.

Component interaction

- **Parent to Child—Use Prop**
  - If you have access to data your children component need from the parent component, all you need is to pass it as a prop to the child.
- **Child to Parent—Use a callback and states**
  - Define a callback in parent which takes the data as a parameter.
  - Pass that callback as a prop to the child
  - Call the callback using this.props.[callback] in the child, and pass in the data as the argument.

Continue...

- **Between Siblings**
    - To pass data between siblings, you have to use the parent as an intermediary. First pass the data from the child to the parent, as an argument into a callback from the parent.
    - Set this incoming parameter as a state on the parent component, then pass it as a prop to the other child. The sibling can then use the data as a prop.

Life cycle methods

- A component has 3 phase
- 1. Initialization
- 2. Update
- 3. Unmount (destruction)

Continue...

- **Initialization phase**
  - getDefaultProps
  - getInitialState
  - componentWillMount
  - Render
  - componentDidMount
- **Update phase**
  - shouldComponentUpdate
  - componentWillUpdate
  - render
  - componentDidUpdate
- **Umounting phase (Destruction)**
  - componentWillUnmount
  - componentDidUnMount

fetch

- **Using fetch to get data from an API**

```
fetch(url)
// Call the fetch function passing the url of the API as a parameter
.then(function() {
 // Your code for handling the data you get from the API
}) .catch(function() {
// This is where you run code if the server returns any errors
});
```

```
fetch('http://localhost:9292/RestDemo/rest/obj/passObj', {

        method: 'POST',
         headers: {
         Accept: 'text/plain',
        'Content-Type': 'application/json',
        },
         body: JSON.stringify({
        id:this.state.id,
        name:this.state.name,
        salary:this.state.salary}),}).
        then(function(res){
                      return res.text();}).
        then(function(result){
                     console.log(result)
        })
```

Axios service

- Axios is a Promise-based HTTP client for JavaScript which can be used in your front-end application and in Rest API.
- By using Axios it's easy to send asynchronous HTTP request to REST endpoints and perform CRUD operations.
- The Axios library can be used in your plain JavaScript application

Continue...

- To install the axios plugin
- npm install –g axios
- Get method
  - axios.get(url).then()
- Post method
  - axios.post(url,data).then()

Sometimes we Need JavaScript Functions

- Use **map()** to apply a function to an array of items:
  - Returns a new array with ‘transformed’ elements
- You specify the function
  - Use **filter()** to return a subarray of items: involves conditional logic (defined by you)
  - Other functions: **merge(), flatten(), reduce()**, …
  - you can combine them via method chaining

React Native

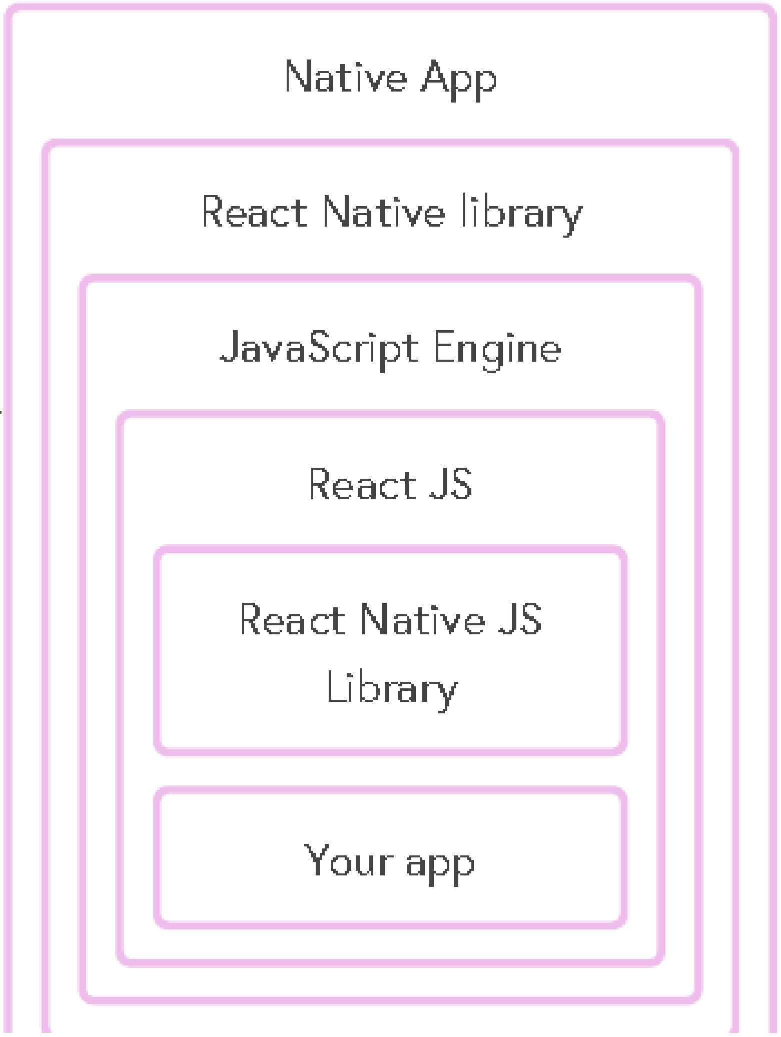
Hybrid API

- A web view
- Running a web app
- With access to native APIs
- That you build once and deploy everywhere.

React Native Is An Extension of Hybrid

What Is It?

- An embedded instance of **JavaScriptCore**.
- React components with bindings to native UI components.
- Manipulating calls into Objective C & Java for behavior.
- And polyfills for some web APIs



Getting started

- To install the react native
  - npm install -g create-react-native-app
- To create the project
  - create-react-native-app demo-app
  - cd demo-app
  - npm start
- To run in device
  - npm run android or npm run ios

Run the project with live reload

- Install the watch man which help to live reload.
  - npm install -g watchman
- Then we can execute the command with watch man
  - npm run android -watchman

React native components

- View
- Text
- ListView
- ScrollView
- TextInput
- Navigator
- Image
- ……

TextInput

- <TextInput
  ref="title"
  autoFocus={true}
  placeholder={Untitled}
  style={styles.title} />
- <TextInput
  ref="description"
  multiline={true}
  placeholder={Description}
  style={styles.description} />

Button

- <TouchableOpacity onPress={() => console.log('pressed')}>

  <View>

  <Text>Button</Text>

  </View>

  </TouchableOpacity>

Styles and Layout

```
const styles = StyleSheet.create({
container: {
flex: 1,
justifyContent: 'center',
alignItems: 'center',
backgroundColor: '#F5FCFF',
},
welcome: {
fontSize: 20,
textAlign: 'center',
},
instructions: {
textAlign: 'center',
marginBottom: 5,
},});
```

Navigator

```
<Navigator
initialRoute={{name: 'My First Scene', index:
0}}
renderScene={(route, navigator) =>
// Return view based on route
} />
```

Navigator Methods

- getCurrentRoutes() – returns the current list of routes
- jumpBack() – Jump backward without unmounting the current scene
- jumpForward() – Jump forward to the next scene in the route stack
- jumpTo(route) – Transition to an existing scene without unmounting
- push(route) – Navigate forward to a new scene
- pop() – Transition back and unmount the current scene
- replace(route) – Replace the current scene with a new route
- replaceAtIndex(route, index) – Replace scene specified by index
- replacePrevious(route) – Replace the previous scene
- resetTo(route) – Navigate to a new scene and reset route stack

Fetch data

```
fetch(encodeURI('http://myendpoint'))
then(response => {
// Handle data
});
```

Styles

- marginTop, marginLeft,marginRigh,marginBottom
- backgroundColor
- borderWith,borderHeight
- fontWeight,fontSize

The github documentation for styles

- https://github.com/vhpoet/react-native-styling-cheat-sheet/blob/master/README.md

React Native component official documentation

- https://facebook.github.io/react-native/docs

app.json file

```
{ "expo":
  { "name": "your app name",
   "description": "your app desc",
        "ios": { "supportsTablet": true },
        "android":
                { "package": "com.abc.demo" }
   }
 }
```

Create APK for android

- npm install –g exp
-
- exp build:android
- exp build:ios
- exp build:status