

The background of the image is a light blue color with various abstract shapes. There are several white circles of different sizes, some with red outlines. There are also white lines and shapes that resemble a stylized globe or a network diagram. In the center, there is a white smartphone outline. Inside the smartphone, there is a search bar with a magnifying glass icon on the left and a red 'x' icon on the right. Below the search bar, there are some faint, stylized shapes that look like a list or a grid of items.

React Native

Ayyappan Murugesan

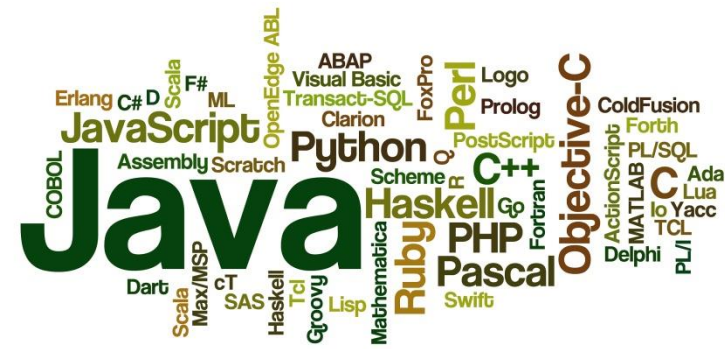
Call: 7010774329

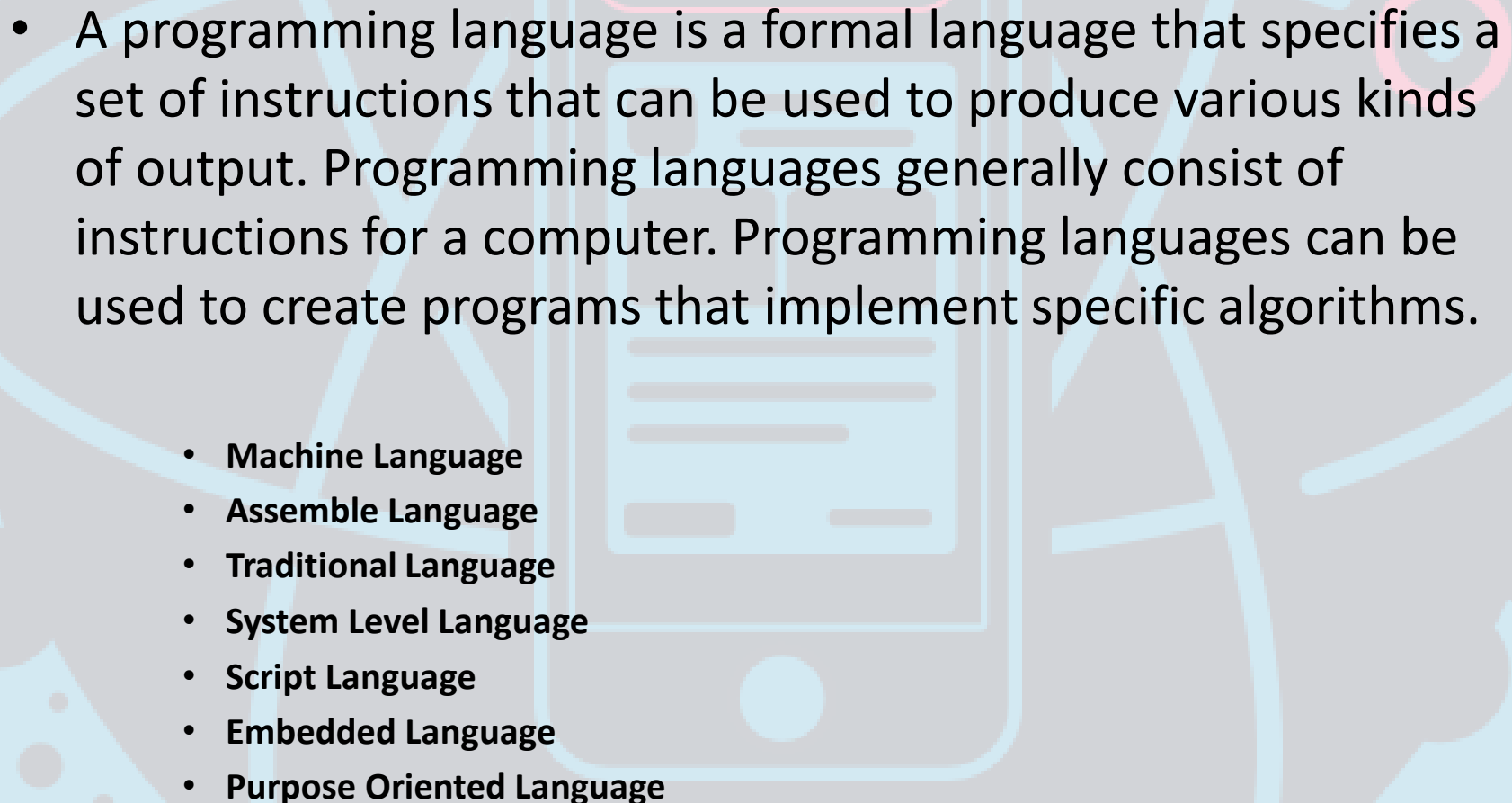
WhatsApp: 8870483557

Session 1

- Programming Language
- Web & Mobile Development
- What is Native Development
- Why JavaScript?
- React Native Intro
- React Native Installation
- Training Agenda

Programming Language



- 
- A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.
- **Machine Language**
 - **Assemble Language**
 - **Traditional Language**
 - **System Level Language**
 - **Script Language**
 - **Embedded Language**
 - **Purpose Oriented Language**

Web & Mobile Development

- Web development broadly refers to the tasks associated with developing websites for hosting via intranet or internet. The web development process includes
 - **Client Side Development (Front End)**
 - **Server Side Development (Back End)**
- Mobile application development is the set of processes and procedures involved in writing software for small, wireless computing devices such as smartphones or tablets.
 - **Android**
 - **iOS**
 - **Windows**

What is Native Development?

- A **native** application is a software program that is developed for use on a particular platform or device
- **Native** apps can provide optimized performance
 - Native Development Kit (NDK)
 - Objective C

Why JavaScript ?

- Dynamic
- jQuery
- Model View Controller
- Backbone JS
- Angular JS
- React JS
- **Node JS**

Cross Platform Native Development


React Native

- Build native mobile apps using JavaScript and React
- A React Native app is a real mobile app
- Don't waste time recompiling
- React Native combines smoothly with components written in Objective-C, Java, or Swift
- Facebook Team

Platforms



Installation

- Setup Chocolatey (choco) 
 - <https://chocolatey.org/install>
- Install Node JS, Python2 & JDK
 - choco install -y nodejs.install python2 jdk8
- Install React Native Cli
 - npm install -g react-native-cli
- Install Android SDK / Xcode SDK

Upcoming Session

- Basics (JSX , Components, State , Props)
- Styling , Flexbox, Dimension
- Default Components
View,Text,Button,ListView,ScrollView
WebView,Images,Picker,Status Bar
Switch,Alert,Modal
- Advance Concept
Async Storage,HTTP,Camera Roll,Geo Location
Router

Session 2

- Example 001 → HTML View
- Example 002 → JavaScript Controller
- Example 003 → JS Business Logic
- Example 004 → JS View ↔ Controller
- Example 005 → VC using jQuery
- Example 006 → MVC using Backbone
- Example 007 → JSX Intro
- Example 008 → React JS Prop
- Example 009 → React JS State
- Example 010 → MVC using React JS
- Create Project using `react-native cli`
- Create Project using `create-react-native-app`

Example 001 → HTML View

- HyperText Markup Language
- HyperText → Formatting Information
- Markup → Tags → `<....>` `</....>`
- Never Respond to User Actions
- Static
- Globally it is called VIEW

Example 002 → JavaScript Controller

- `<script>` Tag
- Responding to User Inputs
- Common Objects
 - Window → Browser Object
 - Document → Page Object
 - `addEventListener` → Mapping User Action
 - Console → Browser debugging log
- User Action → Event → Controller

Example 003 → JS Business Logic

- **Get View elements**
- **Retrieving data from View**
- **Performing Business Logic**

`<script>`

```
var textInputElement = document.getElementById('textInput');
```

```
textInputElement.addEventListener('keyup', function(){
```

```
  var text = textInputElement.value;
```

```
  console.log('New text is "' + text + '"');
```

```
});
```

```
</script>
```

Example 004

→ JS View ↔ Controller

- **Construct data from Business Logic**
- **Set data to View Element**
- **Dynamic View**
- **Syncing View & Controller**

```
<script>  
  var textInputElement = document.getElementById('textInput'),  
      nameDivElement = document.getElementById('nameDiv');  
  
  textInputElement.addEventListener('keyup', function(){  
    var text = textInputElement.value;  
    nameDivElement.innerHTML = text;  
  });  
</script>
```

Example 005 → VC using jQuery

- Simplify Syntax
- Higher Performance
- Getting element with **\$**
- **Model**

Name: <input id="textInput" type="text"/>
Hello !

```
<script>  
    $('#textInput').on('keyup', function(){  
        $('#nameDiv').html($('#textInput').val());  
    });  
</script>
```


Example 006 → MVC using Backbone

// Model → Storing Data

```
var model = new Backbone.Model({  
  name: ''  
});
```

// View → UI

```
model.on('change:name', function(){  
  $('#nameSpan').html(model.get('name'));  
});
```


// Controller → User Action

```
$('#textInput').on('keyup', function(){  
  model.set('name', $('#textInput').val());  
});
```

Example 007 → JSX Intro

- Babel JS - JavaScript compiler
- Simplify the Custom Component Creation
- Inject HTML without Stringing
- Certain format is different from the HTML
 - Style
 - className ... etc

Example 008 → React JS Prop

- Custom Element Property 
- User can sync @ any time
- Navigation Properties
- Also carry children information
- Primary parameter for functiona and React.Component class

Example 009 → React JS State

- Equal to **Model**
- State available when the component is alive
- State has some sequence flow
- `{}` → Embed the state into **View**
- `setState` → Update data from **Controller**

Mounting State

- constructor()
- static getDerivedStateFromProps()
- componentWillMount() / UNSAFE_componentWillMount()
- render()
- componentDidMount()

Updating

- `componentWillReceiveProps()` / `UNSAFE_componentWillReceiveProps()`
- `static getDerivedStateFromProps()`
- `shouldComponentUpdate()`
- `componentWillUpdate()` / `UNSAFE_componentWillUpdate()`
- `render()`
- `getSnapshotBeforeUpdate()`
- `componentDidUpdate()`

Unmounting

- `componentWillUnmount()` 



Example 010 → MVC using React JS

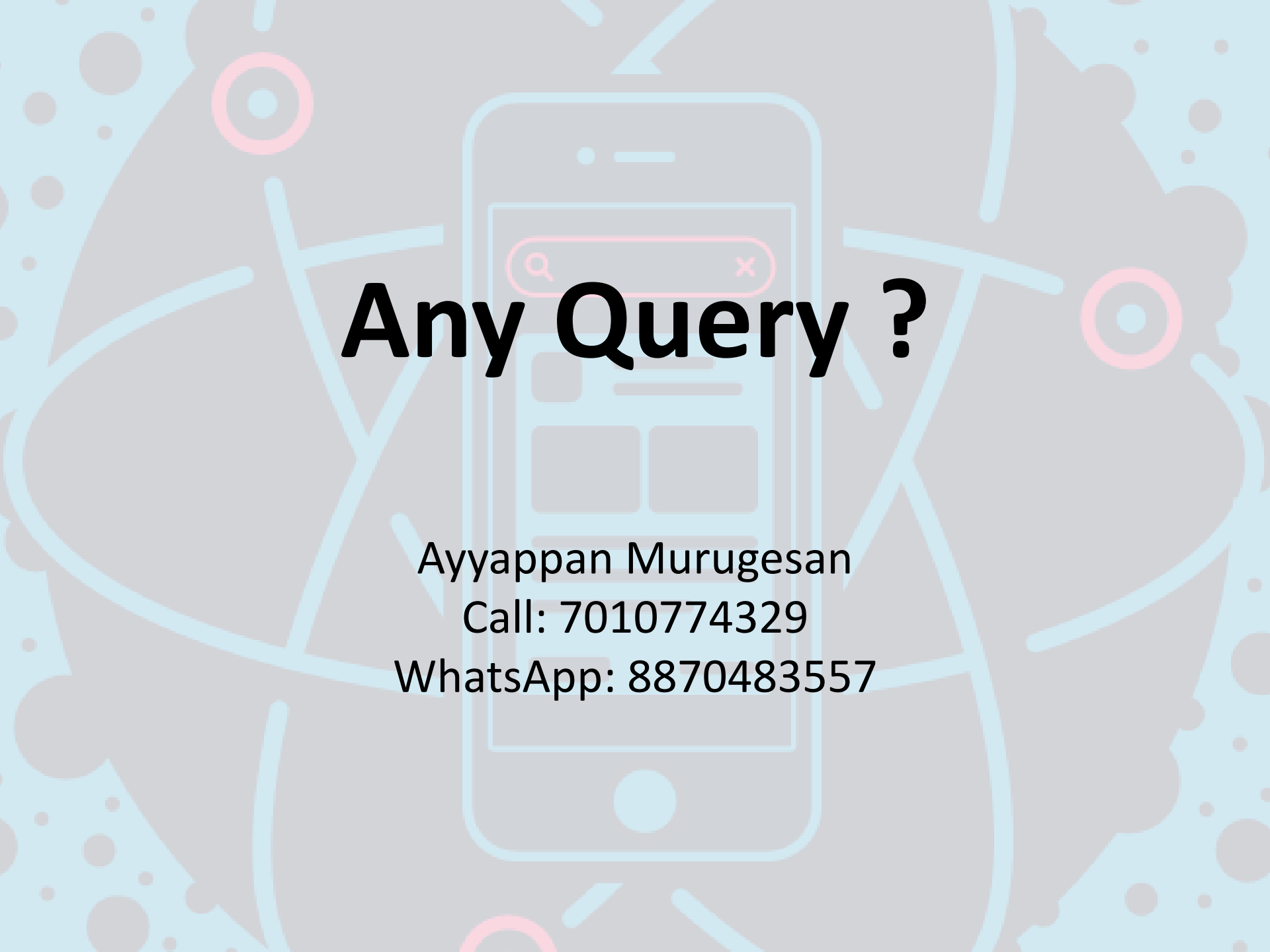
```
class MyText extends React.Component {  
  constructor(props) {  
    super(props);  
    // Model  
    this.state = {  
      text: ""  
    };  
  }  
  // Controller  
  handleChange(event) {  
    this.setState({text: event.target.value});  
  }  
  // View  
  render() {  
    return (  
      <div>  
        <input type="text" value={this.state.text} onChange={(e) => this.handleChange(e)} />  
        <h1>Hello, {this.state.text}</h1>  
      </div>  
    );  
  }  
}
```


Create Project using react-native cli

- react-native init **MyProject**
- **Connect Your Debugging Mobile / Emulator**
- react-native run-android / run-ios
- react-native log-android
- react-native start
- **Note: Your Mobile and Machine in same network**

Create Project using create-react-native-app

- Install Expo on your mobile
- `npm install -g create-react-native-app`
- `create-react-native-app MyProject`
- `cd AwesomeProject`
- `npm start`
- Scan QR Code into Your Mobile
- **Note: Your Mobile and Machine in same network**

The background of the slide features a central illustration of a smartphone. On the screen of the phone, there is a search bar with a magnifying glass icon on the left and a close 'x' icon on the right. Below the search bar, there are several rectangular blocks representing app icons or search results. The entire scene is set against a light blue background decorated with abstract white and pinkish-red shapes, including circles, lines, and a large, faint outline of a person's head and shoulders.

Any Query ?

Ayyappan Murugesan

Call: 7010774329

WhatsApp: 8870483557