

# **THEMING REACT NATIVE COMPONENTS**

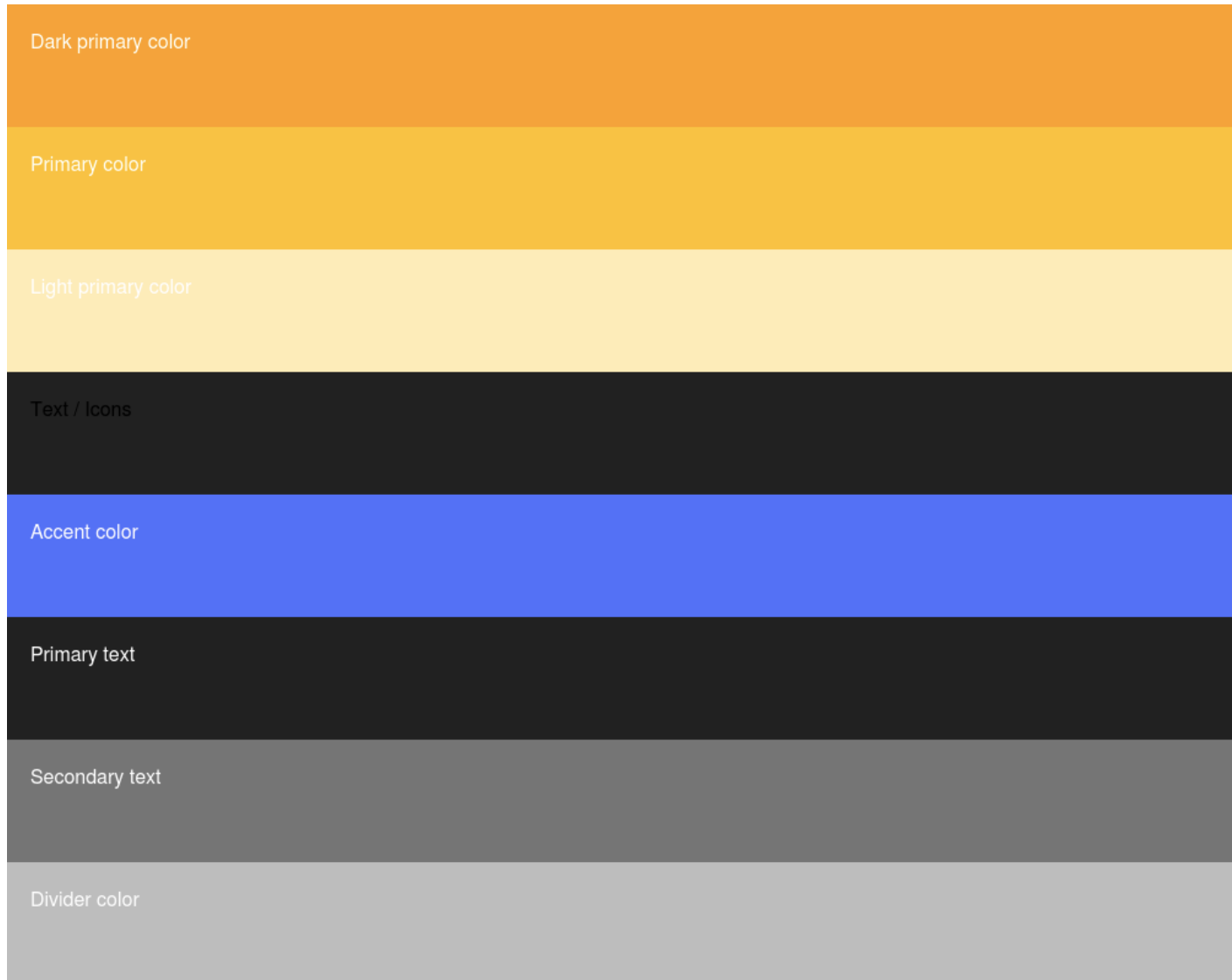
**WHO AM I?**

Binoy Patel

[binoy.io](http://binoy.io)

[@binoypl4](https://twitter.com/binoypl4)

# Color Palette



# USING REACT NATIVE ELEMENTS

```
<Button  
  color="#212121"  
  backgroundColor="#FFC107"  
  title="Press Me!"  
  borderRadius={4}  
  buttonStyle={{ borderColor: '#FFA000', borderWidth: 1}}  
>
```

# Component Preview

Fancy Card	
List 2 I am a subtitle	>
List 3 I am a subtitle	>
List 4 I am a subtitle	>
List 5 I am a subtitle	>
Press Me!	

# IMPLEMENTATION

```
<Card
  title="Fancy Card"
  containerStyle={{ paddingHorizontal: 0, borderColor: "#BdBDBD" }}
  dividerStyle={{ borderColor: "#BdBDBD" }}
  titleStyle={{ color: "#212121" }}
>
  <ListItem
    title="List 2"
    subtitle="I am a subtitle"
    titleStyle={{ color: '#212121' }}
    subtitleStyle={{color: '#757575'}}
  />
  <Divider style={{ backgroundColor: '#BdBDBD' }}/>
  <ListItem
    title="List 3"
    subtitle="I am a subtitle"
    titleStyle={{ color: '#212121' }}
    subtitleStyle={{color: '#757575'}}
  />
  <Divider style={{ backgroundColor: '#BdBDBD' }}/>
  <Button
    color="#212121"
    backgroundColor="#FFC107"
    title="Press Me!"
    borderRadius={4}
    buttonStyle={{ borderWidth: 1, marginTop: 10}}
  />
</Card>
```

**SUCH CODE REUSE**

**MUCH DRY. WOW.**

[memegenerator.net](http://memegenerator.net)

# With Theming Support

```
const theme = {
  dividerColor: '#BDBDBD',
  primaryTextColor: '#212121',
  secondaryTextColor: '#757575',
  primaryColor: '#FFC107',
};

<ThemeProvider theme={theme}>
  <Card
    title="Fancy Card"
    containerStyle={{ paddingHorizontal: 0 }}
  >
    <ListItem
      title="List 2"
      subtitle="I am a subtitle"
    />
    <Divider />
    <ListItem
      title="List 3"
      subtitle="I am a subtitle"
    />
    <Divider />
    <Button
      title="Press Me!"
      borderRadius={4}
      buttonStyle={{ borderWidth: 1, marginTop: 10 }}
    />
  </Card>
</ThemeProvider>
```



## **How does it work?**

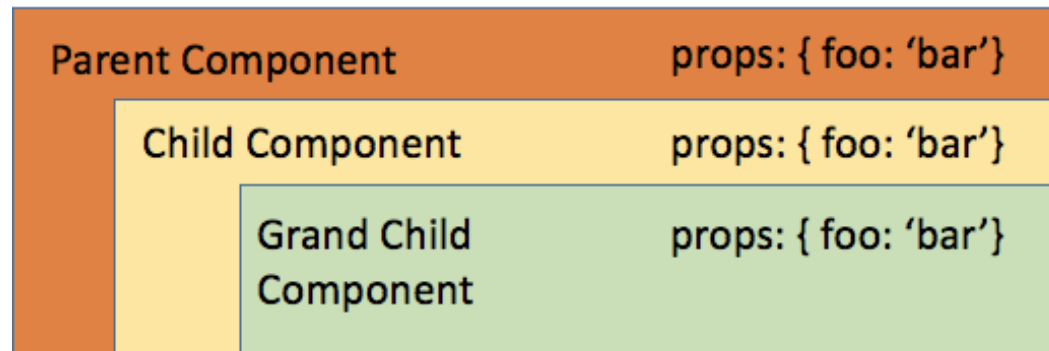
- High Order Components
- Context

# RENDERING THE CHILDREN

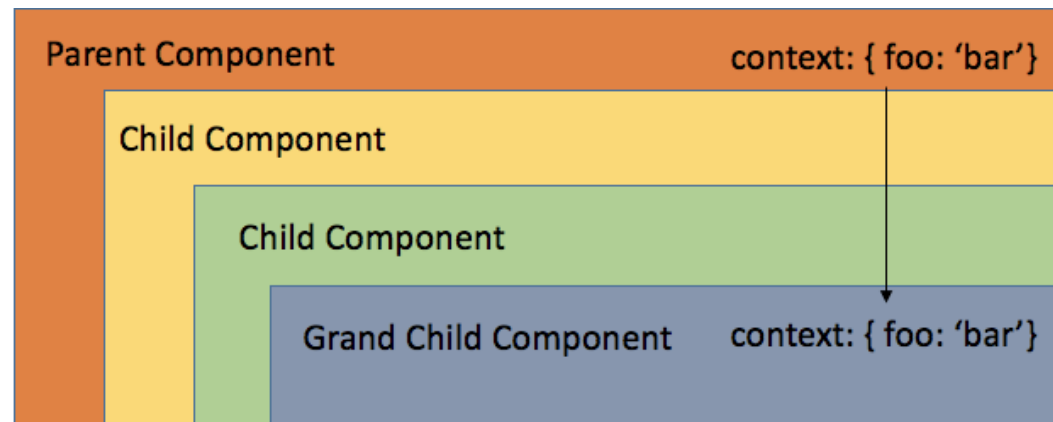
```
export default class ThemeProvider extends Component {  
  render() {  
    const { children } = this.props;  
    const count = React.Children.count(children);  
    if (count === 1) {  
      return React.Children.only(children);  
    } else {  
      return (  
        <View>  
          {children}  
        </View>  
      );  
    }  
  
    return null;  
  }  
}
```

# What is Context

To understand context, first lets take a look at how props are passed to children



Unlike props which need to be passed down by each child. Child components can actually require the context as they need



# Using Context

```
class ParentComponent extends Component {
  getChildContext() {
    return {
      theme: { color: 'blue' }
    }
  }

  static childContextTypes = {
    theme: PropTypes.object
  }
}

export default class GrandChildComponent extends Component {
  static contextTypes = {
    theme: PropTypes.object
  }

  // this.context.theme
  ...
}
```

# Context inside the ThemeProvider

```
export default class ThemeProvider extends Component {  
  getChildContext() {  
    if (this.props.hasOwnProperty('theme')) {  
      const newTheme = {...defaultTheme, ...this.props.theme};  
      return {  
        theme: newTheme,  
      };  
    } else {  
      return {  
        theme: defaultTheme,  
      };  
    }  
  }  
  
  static childContextTypes = {  
    theme: React.PropTypes.object  
  }  
  
  render() {  
    ...  
  }  
}
```

# Comparing different paradigms

## CSS Way in React Native

```
const theme = {
  'library.ui.View': {
    '.h-center': {
      alignItems: 'center',
    },
    '.v-center': {
      justifyContent: 'center',
    },
    '.flexible': {
      flex: 1,
    },
    flexDirection: 'column',
  },
};

class HelloWorld extends Component {
  render() {
    return (
      <StyleProvider style={theme}>
        <View styleName="flexible vertical v-center h-center">
          </View>
        </StyleProvider>
      );
  }
}
```

# Theming Each component

```
const buttonTheme = themeManager.getStyle('Button')

const newButtonTheme = {
  ...buttonTheme,
  BUTTON_FONT_COLOR: '#fff',
  BUTTON_PRIMARY_COLOR: '#2f8cff',
}

themeManager.setSource('Button', () => (newButtonTheme))
```



# React Native Elements Way

```
// Theming whole application
```

```
<ThemeProvider theme={theme}>  
  <App />  
</ThemeProvider>
```

```
// Theming one component
```

```
<ThemeProvider theme={theme}>  
  <Button />  
</ThemeProvider>
```

```
// Theming multiple components separately
```

```
<ThemeProvider theme={lightTheme}>  
  <Button />  
  <Tile />  
</ThemeProvider>  
<ThemeProvider theme={darkTheme}>  
  <Card />  
  <SocialButton />  
</ThemeProvider>
```

## Resources

- [github.com/binoy14/react-native-theming](https://github.com/binoy14/react-native-theming)
- [github.com/react-native-training/react-native-elements](https://github.com/react-native-training/react-native-elements)
- Slides: [binoy14.github.io/RNE-theming-slides](https://binoy14.github.io/RNE-theming-slides)