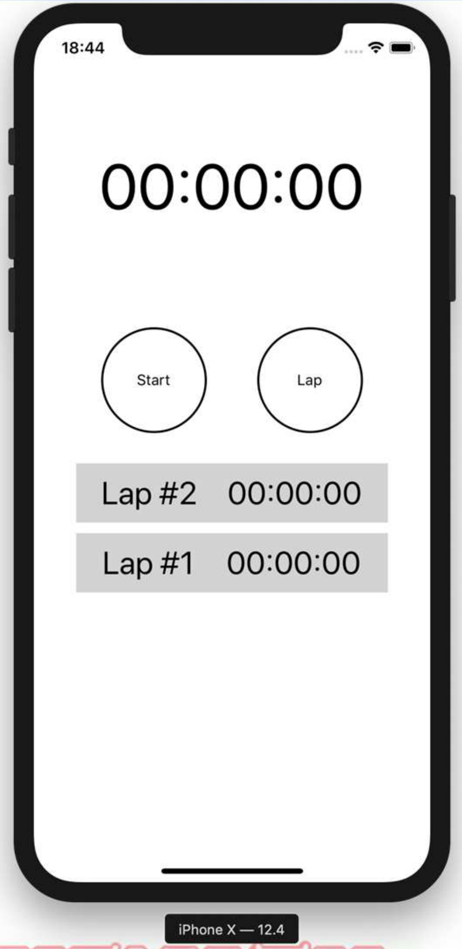
React Native

Timer

StopWatch

StopWatch



00:00:00

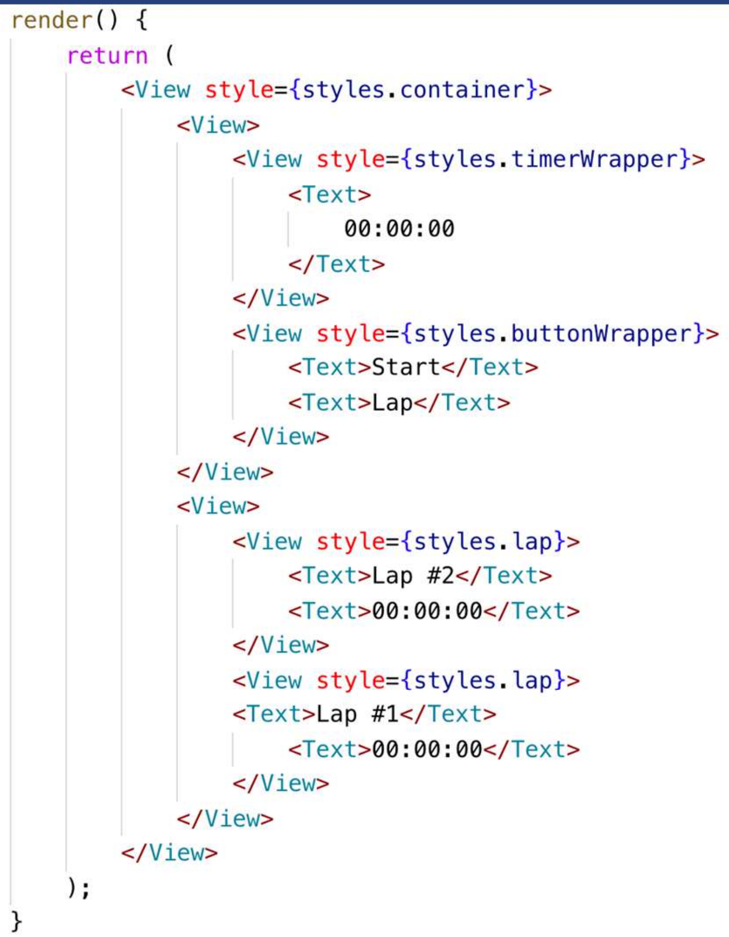
Lap #2 00:00:00

Lap #1 00:00:00

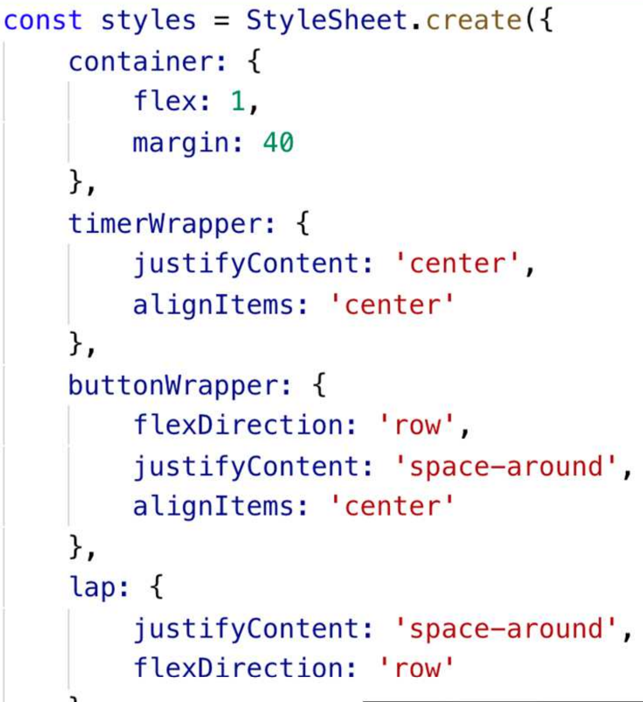
Design App StopWatch includes:

- The timer
  (run and updated each second)
- 2 buttons START/STOP, LAP
- LAP results

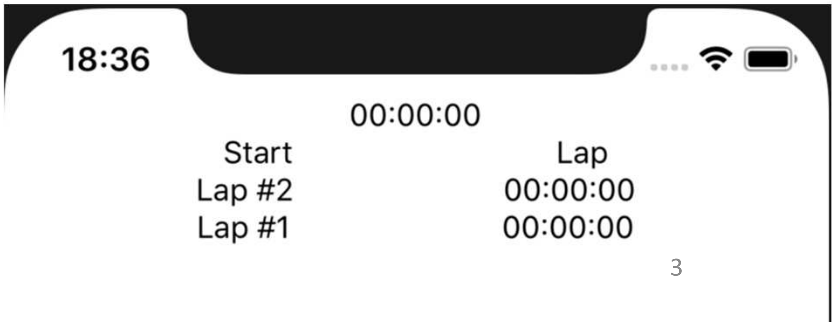
StopWatch



```
render() {
    return (
        <View style={styles.container}>
            <View>
                <View style={styles.timerWrapper}>
                    <Text>
                        00:00:00
                    </Text>
                </View>
                <View style={styles.buttonWrapper}>
                    <Text>Start</Text>
                    <Text>Lap</Text>
                </View>
            </View>
            <View>
                <View style={styles.lap}>
                    <Text>Lap #2</Text>
                    <Text>00:00:00</Text>
                </View>
                <View style={styles.lap}>
                <Text>Lap #1</Text>
                    <Text>00:00:00</Text>
                </View>
            </View>
        </View>
    );
}
```

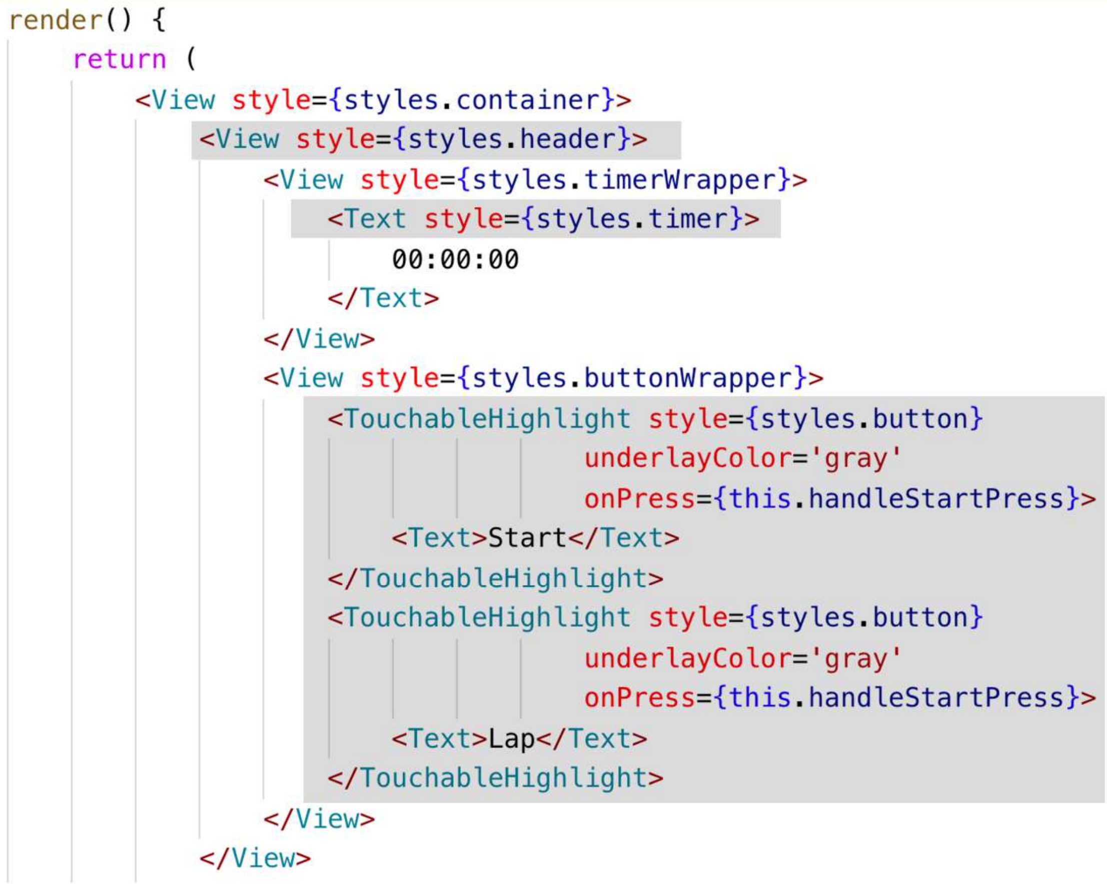


```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        margin: 40
    },
    timerWrapper: {
        justifyContent: 'center',
        alignItems: 'center'
    },
    buttonWrapper: {
        flexDirection: 'row',
        justifyContent: 'space-around',
        alignItems: 'center'
    },
    lap: {
        justifyContent: 'space-around',
        flexDirection: 'row'
    }
});
```

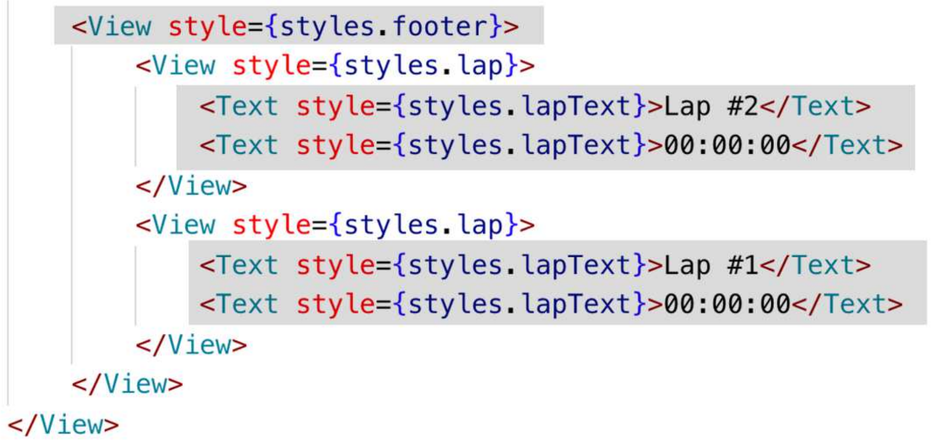


```
              00:00:00
   Start                      Lap
   Lap #2                00:00:00
   Lap #1                00:00:00
```

StopWatch

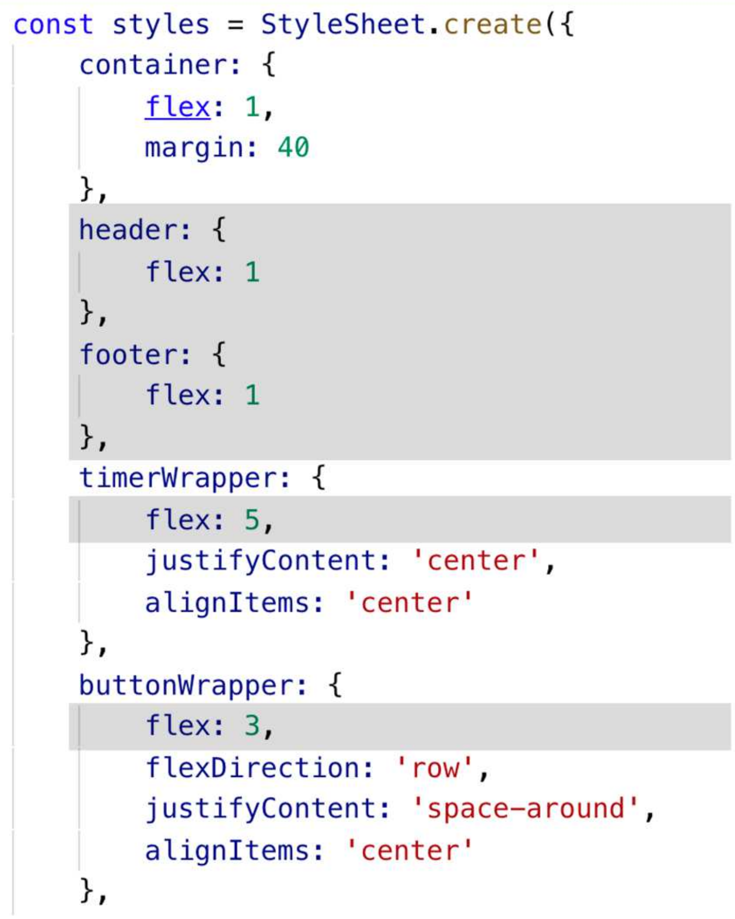


```
render() {
    return (
        <View style={styles.container}>
            <View style={styles.header}>
                <View style={styles.timerWrapper}>
                    <Text style={styles.timer}>
                        00:00:00
                    </Text>
                </View>
                <View style={styles.buttonWrapper}>
                    <TouchableHighlight style={styles.button}
                                        underlayColor='gray'
                                        onPress={this.handleStartPress}>
                        <Text>Start</Text>
                    </TouchableHighlight>
                    <TouchableHighlight style={styles.button}
                                        underlayColor='gray'
                                        onPress={this.handleStartPress}>
                        <Text>Lap</Text>
                    </TouchableHighlight>
                </View>
            </View>
```

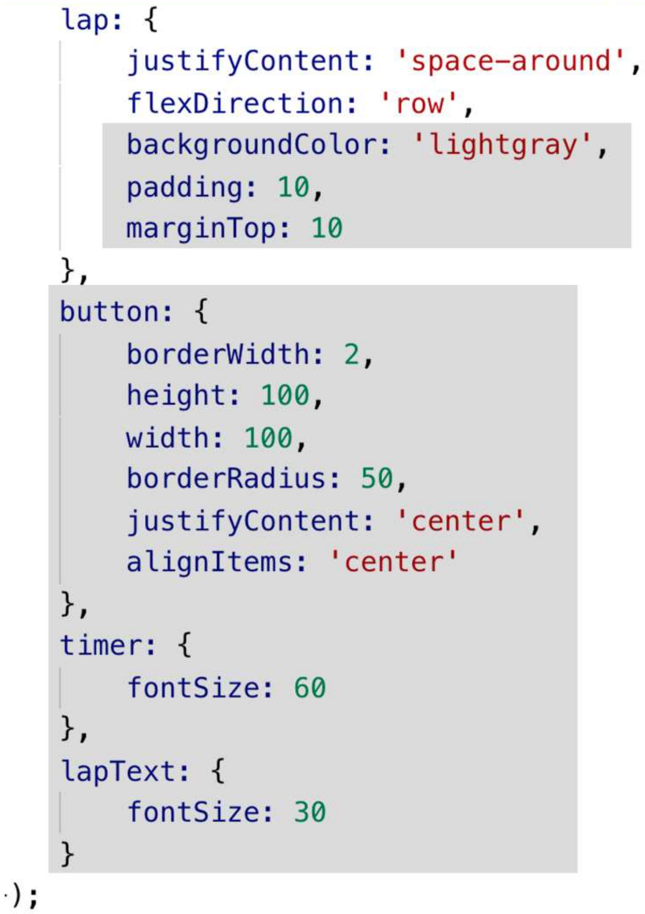


```
    <View style={styles.footer}>
        <View style={styles.lap}>
            <Text style={styles.lapText}>Lap #2</Text>
            <Text style={styles.lapText}>00:00:00</Text>
        </View>
        <View style={styles.lap}>
            <Text style={styles.lapText}>Lap #1</Text>
            <Text style={styles.lapText}>00:00:00</Text>
        </View>
    </View>
</View>
```

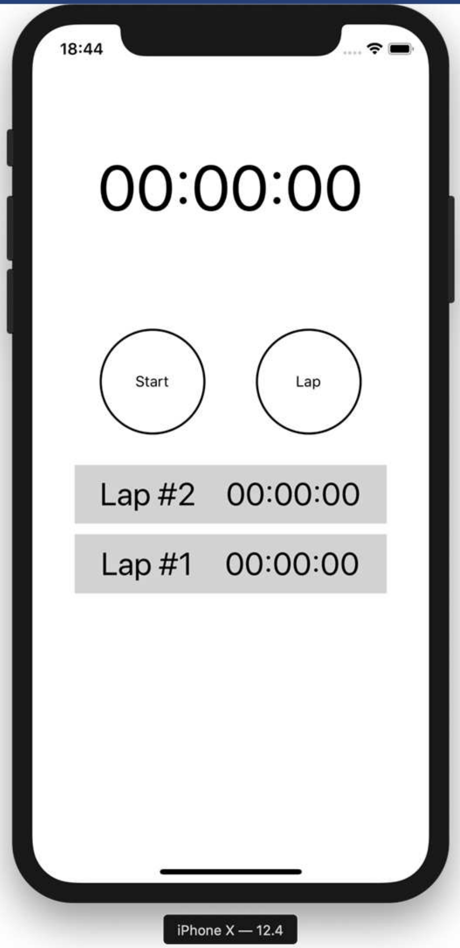
StopWatch



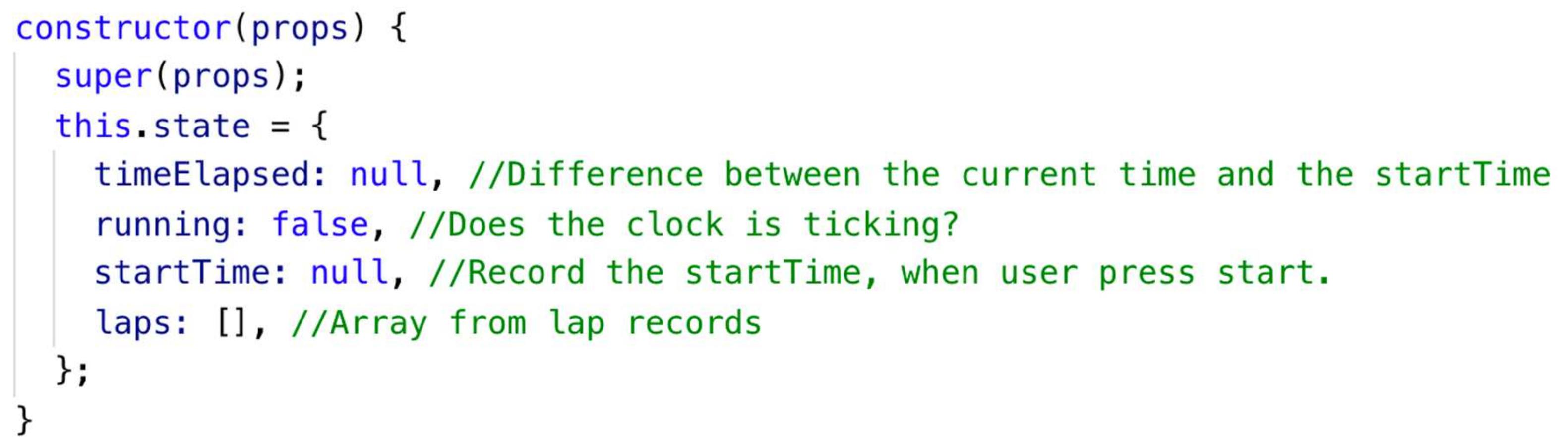
```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        margin: 40
    },
    header: {
        flex: 1
    },
    footer: {
        flex: 1
    },
    timerWrapper: {
        flex: 5,
        justifyContent: 'center',
        alignItems: 'center'
    },
    buttonWrapper: {
        flex: 3,
        flexDirection: 'row',
        justifyContent: 'space-around',
        alignItems: 'center'
    },
```



```
    lap: {
        justifyContent: 'space-around',
        flexDirection: 'row',
        backgroundColor: 'lightgray',
        padding: 10,
        marginTop: 10
    },
    button: {
        borderWidth: 2,
        height: 100,
        width: 100,
        borderRadius: 50,
        justifyContent: 'center',
        alignItems: 'center'
    },
    timer: {
        fontSize: 60
    },
    lapText: {
        fontSize: 30
    }
});
```

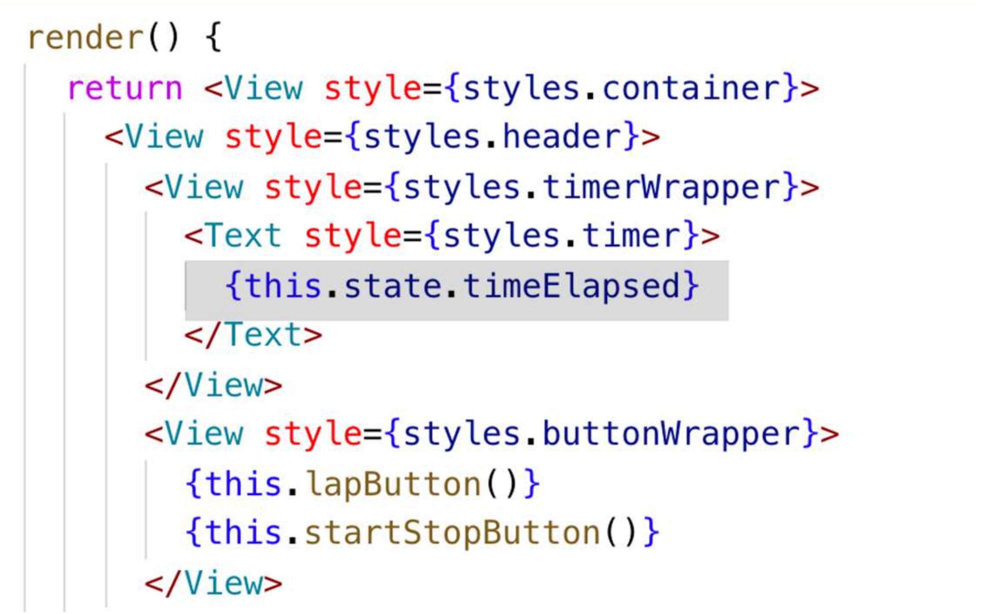


Initialize State



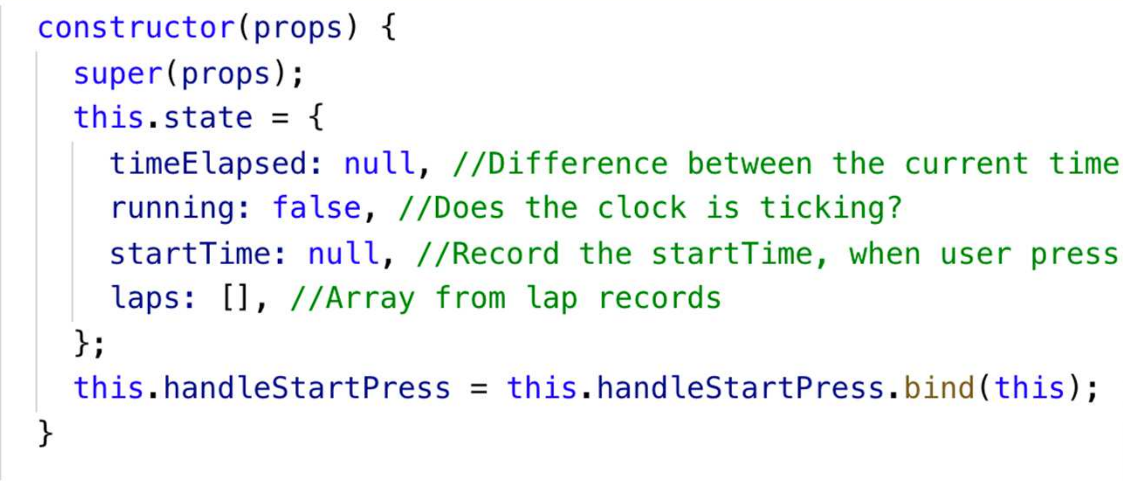
```
constructor(props) {
  super(props);
  this.state = {
    timeElapsed: null, //Difference between the current time and the startTime
    running: false, //Does the clock is ticking?
    startTime: null, //Record the startTime, when user press start.
    laps: [], //Array from lap records
  };
}
```

Showing the TimeElapse

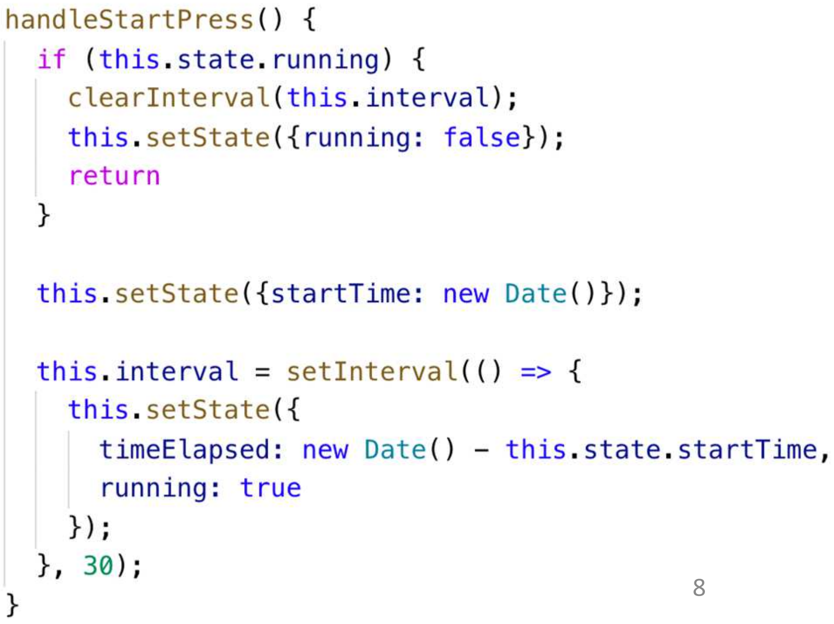


```
render() {
  return <View style={styles.container}>
    <View style={styles.header}>
      <View style={styles.timerWrapper}>
        <Text style={styles.timer}>
          {this.state.timeElapsed}
        </Text>
      </View>
      <View style={styles.buttonWrapper}>
        {this.lapButton()}
        {this.startStopButton()}
      </View>
```

Handle Start Button Pressed



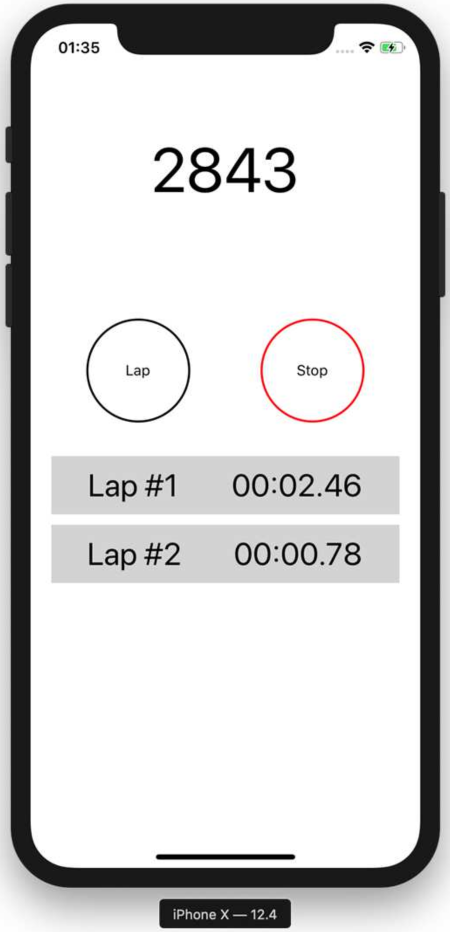
```
constructor(props) {
  super(props);
  this.state = {
    timeElapsed: null, //Difference between the current time and the startTime
    running: false, //Does the clock is ticking?
    startTime: null, //Record the startTime, when user press start.
    laps: [], //Array from lap records
  };
  this.handleStartPress = this.handleStartPress.bind(this);
}
```



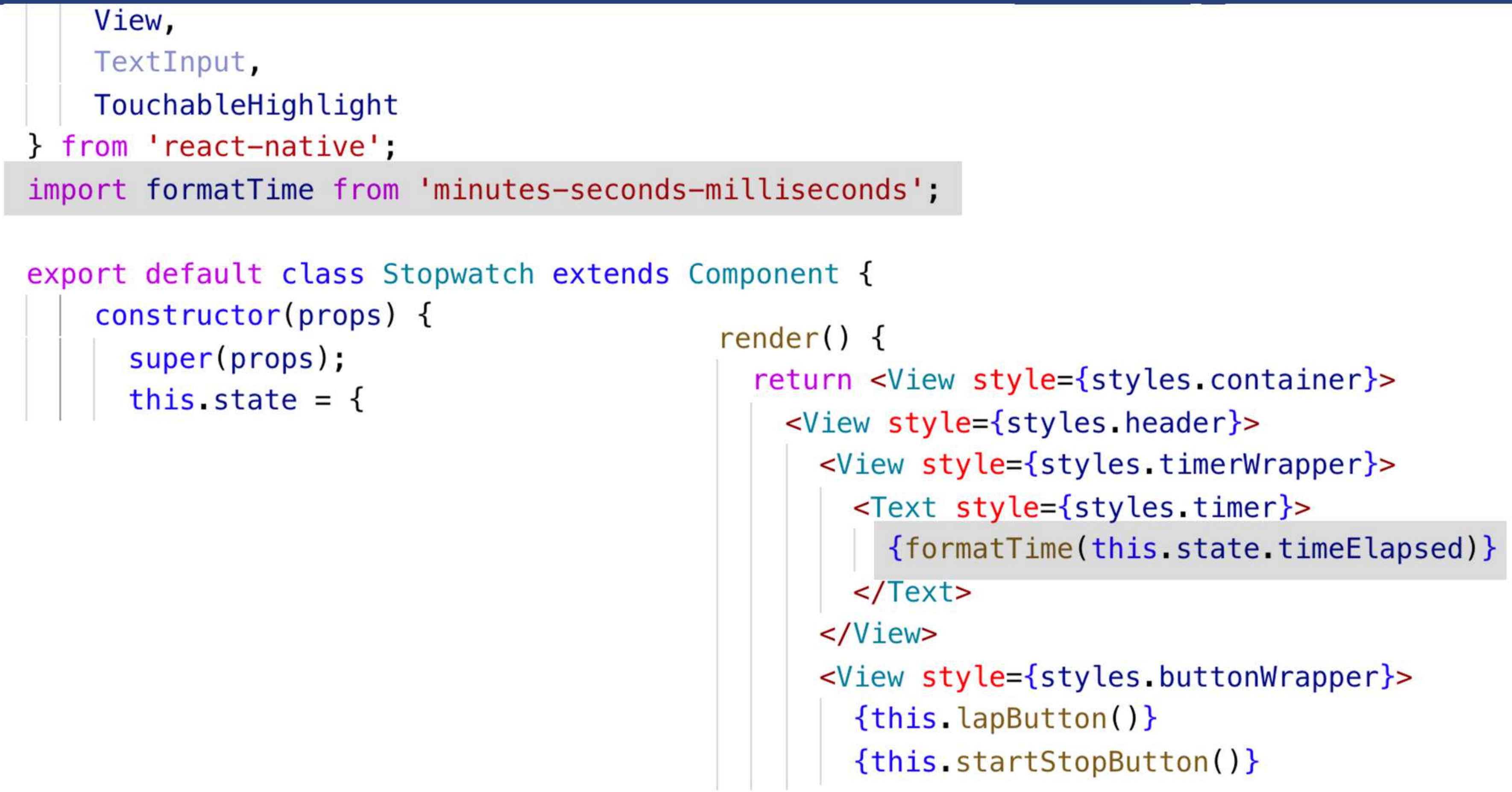
```
handleStartPress() {
  if (this.state.running) {
    clearInterval(this.interval);
    this.setState({running: false});
    return
  }

  this.setState({startTime: new Date()});

  this.interval = setInterval(() => {
    this.setState({
      timeElapsed: new Date() - this.state.startTime,
      running: true
    });
  }, 30);
}
```

Minutes-Second-MilliSeconds Format

- Press Start Button
- TimeElapsed in milliseconds is displayed
- We need to format TimeElapsed to 00:00.00 format
  - We have a library for that. Go to terminal at the working directory
  - `>|$ npm install minutes-seconds-milliseconds --save`



Minutes-Second-MilliSeconds Format

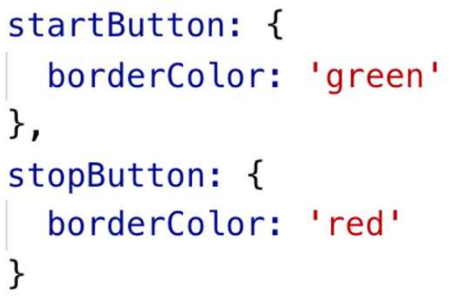


```
    View,
    TextInput,
    TouchableHighlight
} from 'react-native';
import formatTime from 'minutes-seconds-milliseconds';

export default class Stopwatch extends Component {
    constructor(props) {
        super(props);
        this.state = {
```

```
render() {
  return <View style={styles.container}>
    <View style={styles.header}>
      <View style={styles.timerWrapper}>
        <Text style={styles.timer}>
          {formatTime(this.state.timeElapsed)}
        </Text>
      </View>
      <View style={styles.buttonWrapper}>
        {this.lapButton()}
        {this.startStopButton()}
```

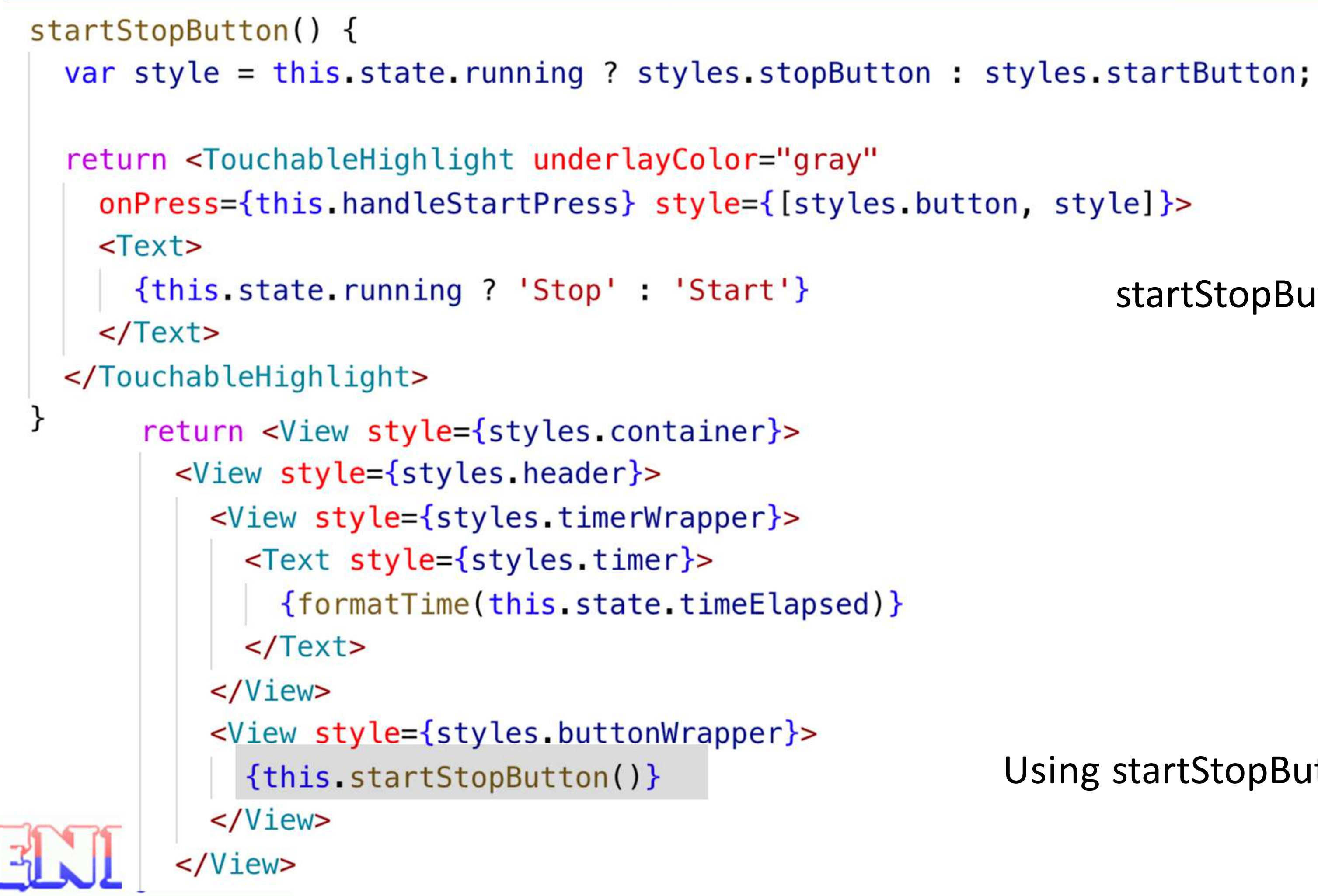
UI Seperation

- We need to make the Stop button first. (Only start is showing now)
- It is a good idea to seperate the start/stop button UI from the main render method, since the main render method is very long (hard to read, and hard to maintenance)



```
startButton: {
  borderColor: 'green'
},
stopButton: {
  borderColor: 'red'
}
```

Start button



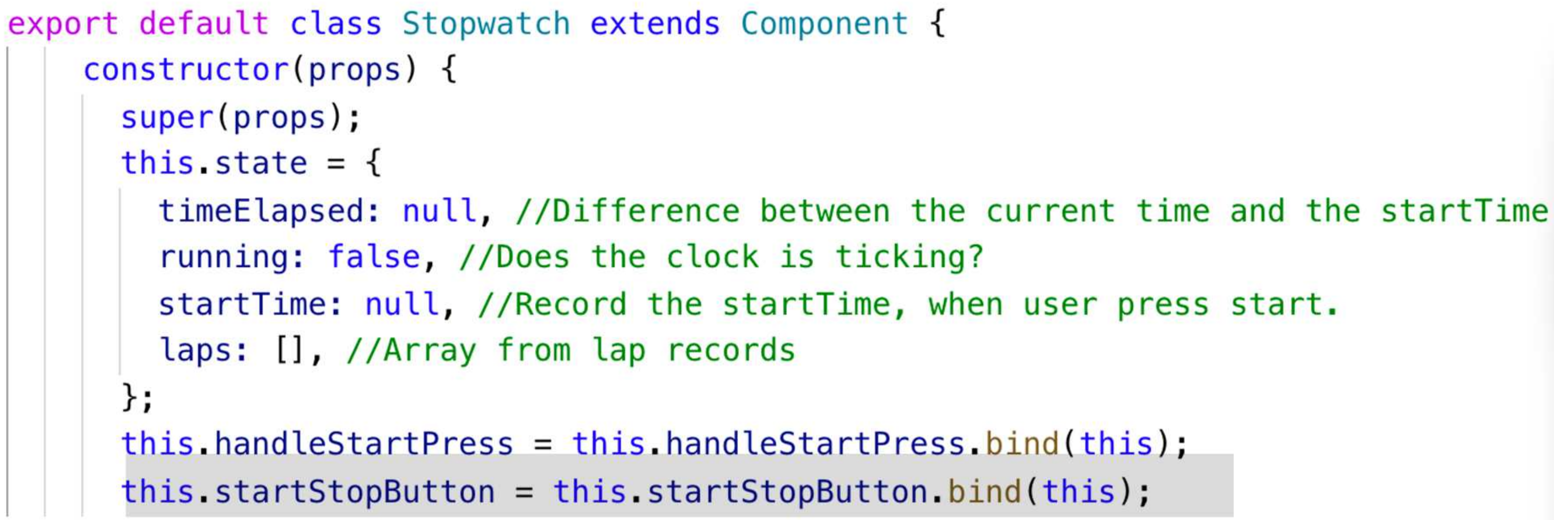
```
startStopButton() {
  var style = this.state.running ? styles.stopButton : styles.startButton;

  return <TouchableHighlight underlayColor="gray"
    onPress={this.handleStartPress} style={[styles.button, style]}>
    <Text>
      {this.state.running ? 'Stop' : 'Start'}
    </Text>
  </TouchableHighlight>
}
```

startStopButton method

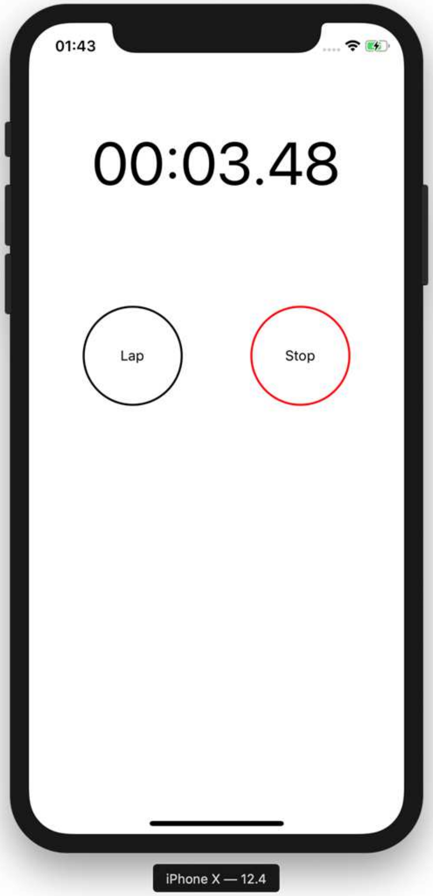
```
return <View style={styles.container}>
  <View style={styles.header}>
    <View style={styles.timerWrapper}>
      <Text style={styles.timer}>
        {formatTime(this.state.timeElapsed)}
      </Text>
    </View>
    <View style={styles.buttonWrapper}>
      {this.startStopButton()}
    </View>
  </View>
```

Using startStopButton method

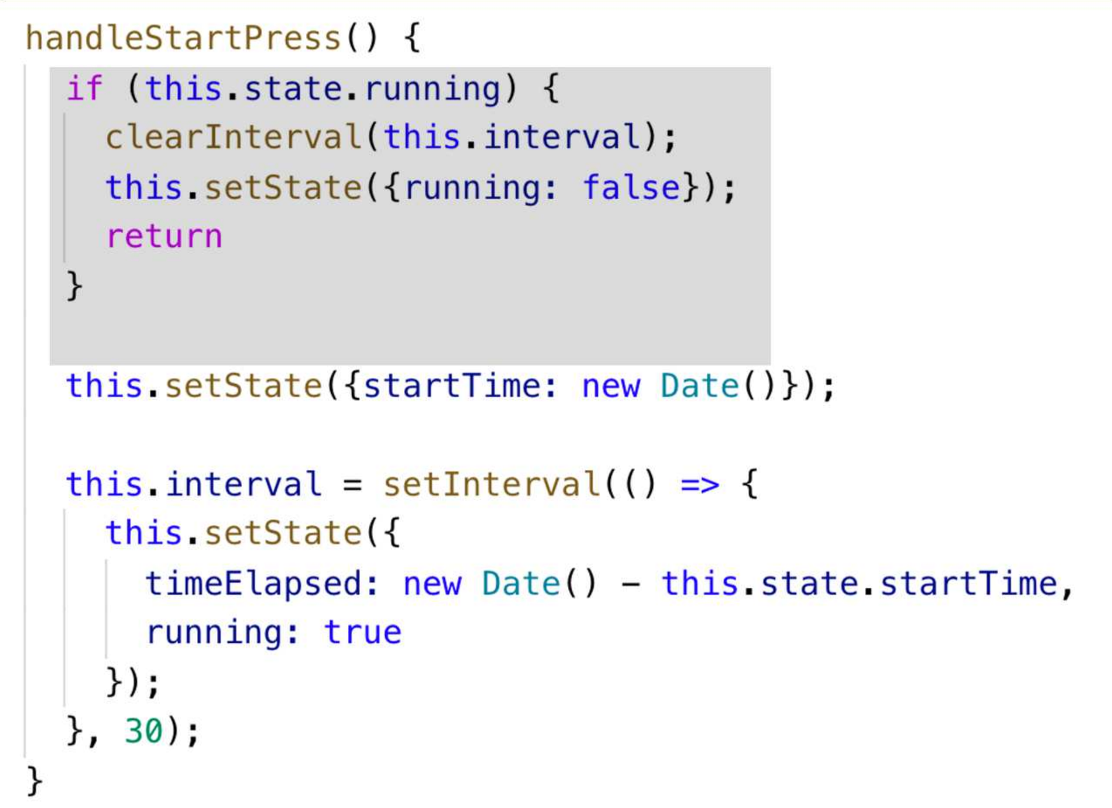
Bind this to StartStopButton



```
export default class Stopwatch extends Component {
    constructor(props) {
      super(props);
      this.state = {
        timeElapsed: null, //Difference between the current time and the startTime
        running: false, //Does the clock is ticking?
        startTime: null, //Record the startTime, when user press start.
        laps: [], //Array from lap records
      };
      this.handleStartPress = this.handleStartPress.bind(this);
      this.startStopButton = this.startStopButton.bind(this);
```

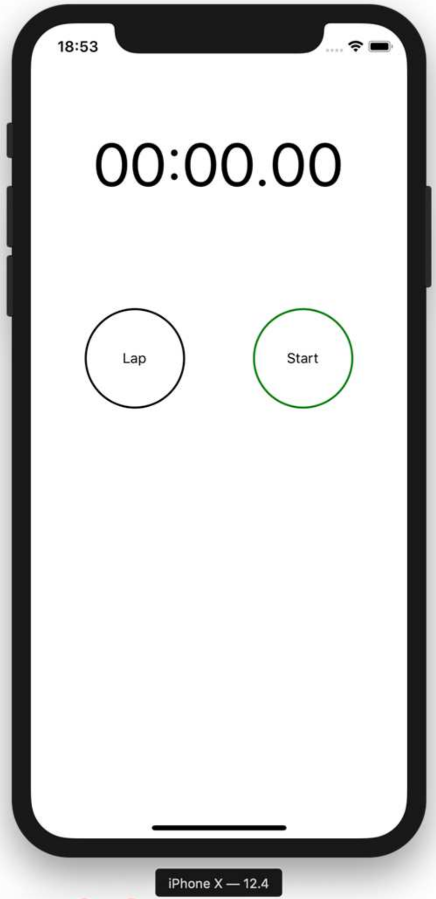


Making timer stop



```
handleStartPress() {
  if (this.state.running) {
    clearInterval(this.interval);
    this.setState({running: false});
    return
  }

  this.setState({startTime: new Date()});

  this.interval = setInterval(() => {
    this.setState({
      timeElapsed: new Date() - this.state.startTime,
      running: true
    });
  }, 30);
}
```

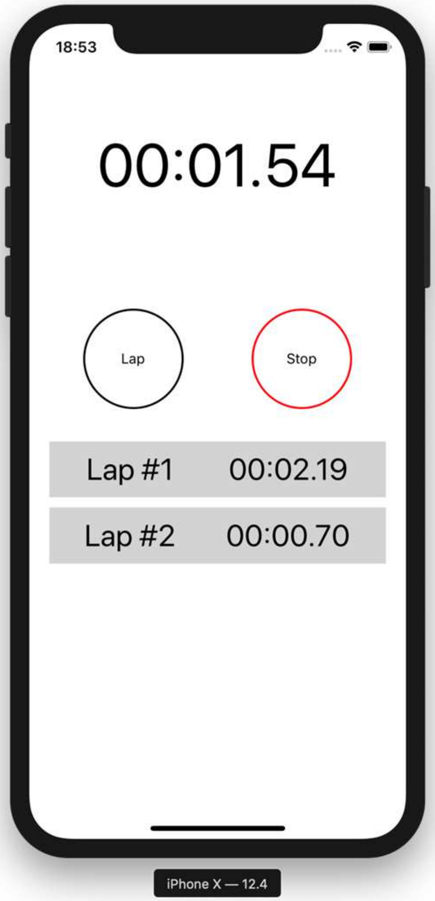
Working start/stop timer app



18:53

00:00.00

iPhone X — 12.4



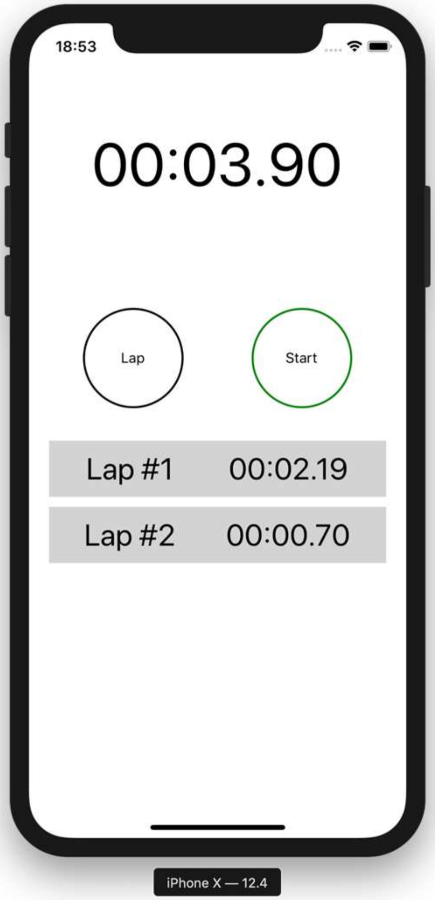
18:53

00:01.54

Lap #1 00:02.19

Lap #2 00:00.70

iPhone X — 12.4



18:53

00:03.90

Lap #1 00:02.19

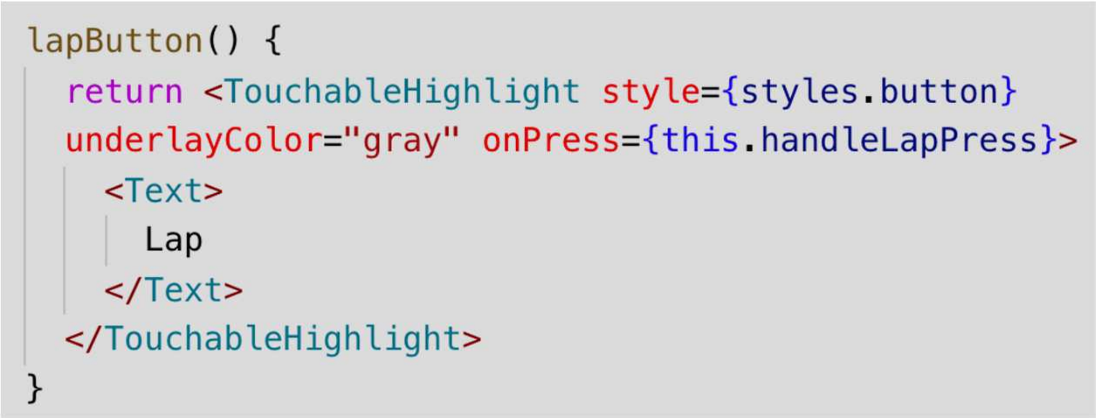
Lap #2 00:00.70

iPhone X — 12.4

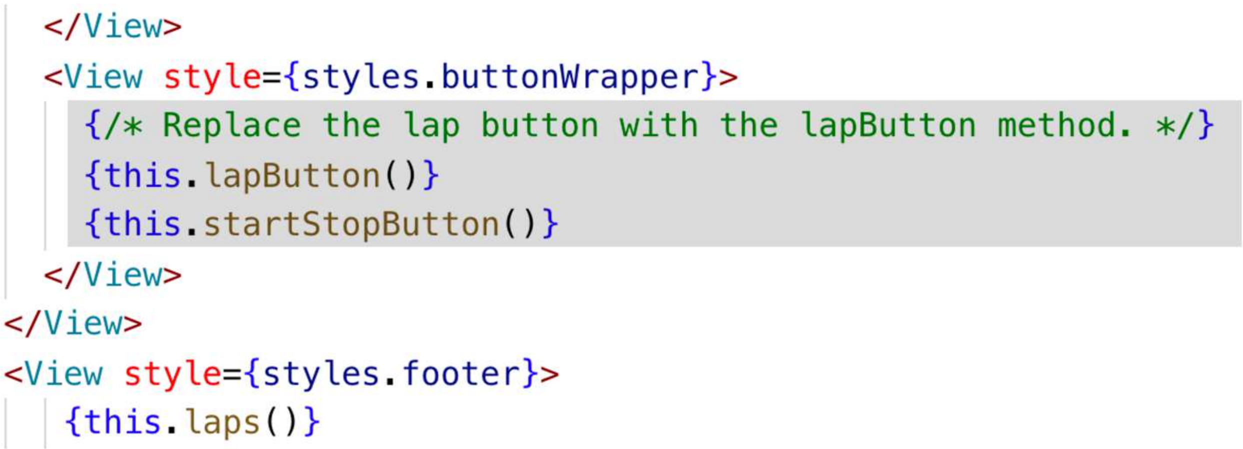
Making lap function works!

- Every time when user press the lap button, record the current timeElapsed into `this.state.lap` and reset the `startTime`
- Make the lower half of the app’s screen showing the list of timeElapsed in `this.state.lap` array.

Lap button UI seperation

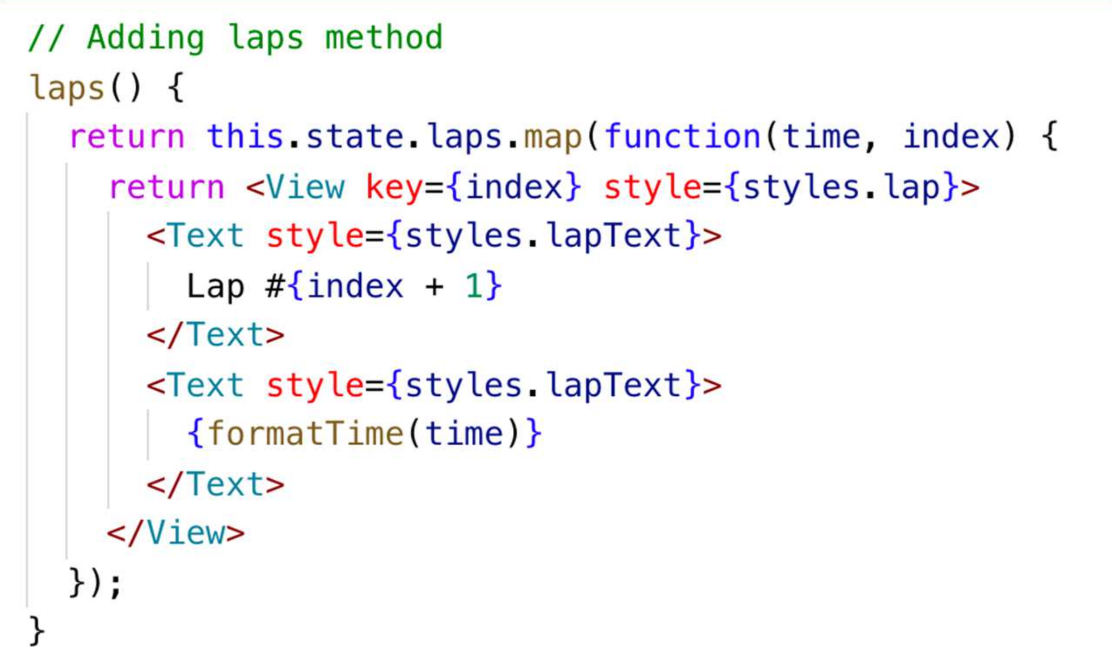


```
lapButton() {
  return <TouchableHighlight style={styles.button}
  underlayColor="gray" onPress={this.handleLapPress}>
    <Text>
      Lap
    </Text>
  </TouchableHighlight>
}
```



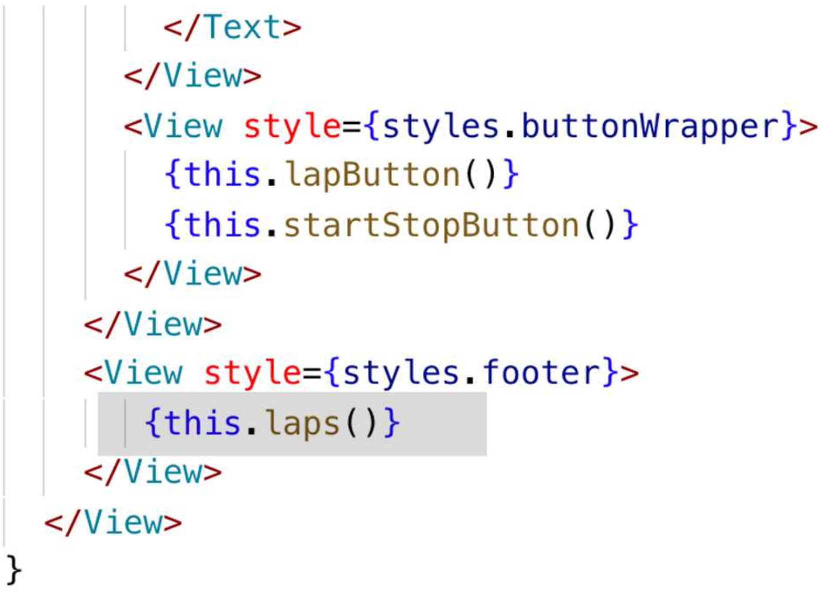
```
  </View>
  <View style={styles.buttonWrapper}>
    {/* Replace the lap button with the lapButton method. */}
    {this.lapButton()}
    {this.startStopButton()}
  </View>
</View>
<View style={styles.footer}>
  {this.laps()}
```

Display the lap array

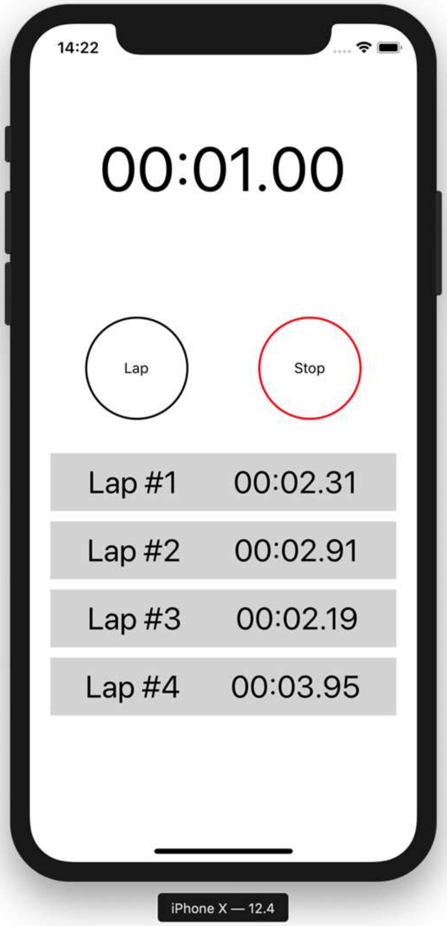


```
// Adding laps method
laps() {
  return this.state.laps.map(function(time, index) {
    return <View key={index} style={styles.lap}>
      <Text style={styles.lapText}>
        Lap #{index + 1}
      </Text>
      <Text style={styles.lapText}>
        {formatTime(time)}
      </Text>
    </View>
  });
}
```

Display the lap array



```
        </Text>
      </View>
      <View style={styles.buttonWrapper}>
        {this.lapButton()}
        {this.startStopButton()}
      </View>
    </View>
    <View style={styles.footer}>
      {this.laps()}
    </View>
  </View>
}
```



00:01.00

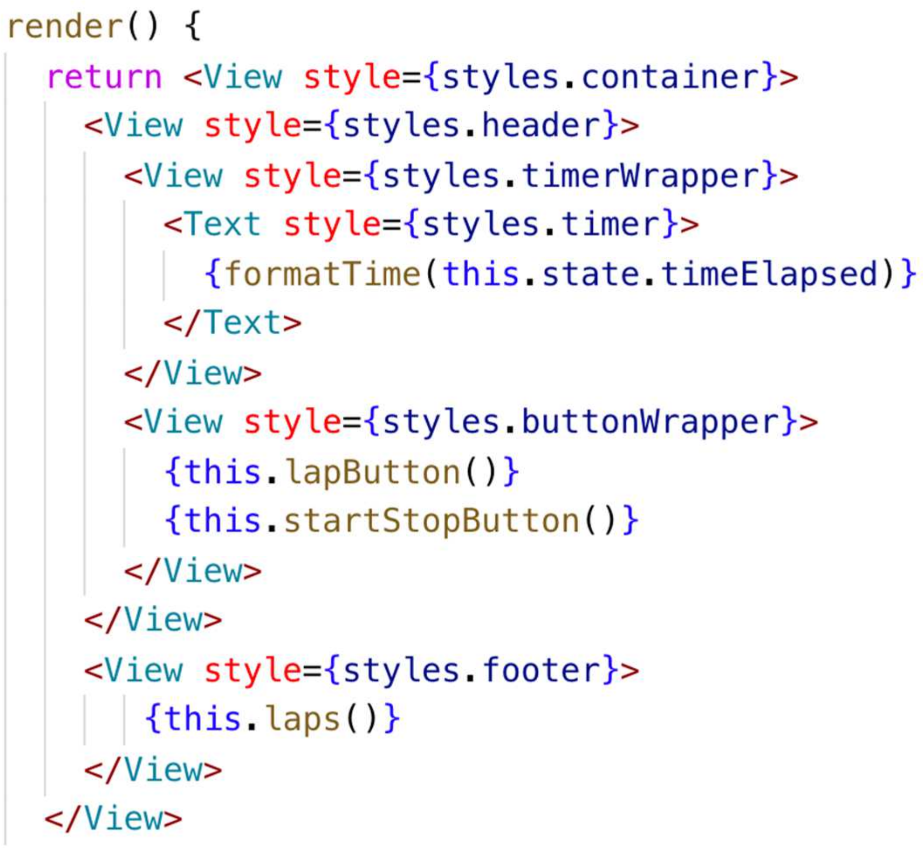
Lap #1 00:02.31

Lap #2 00:02.91

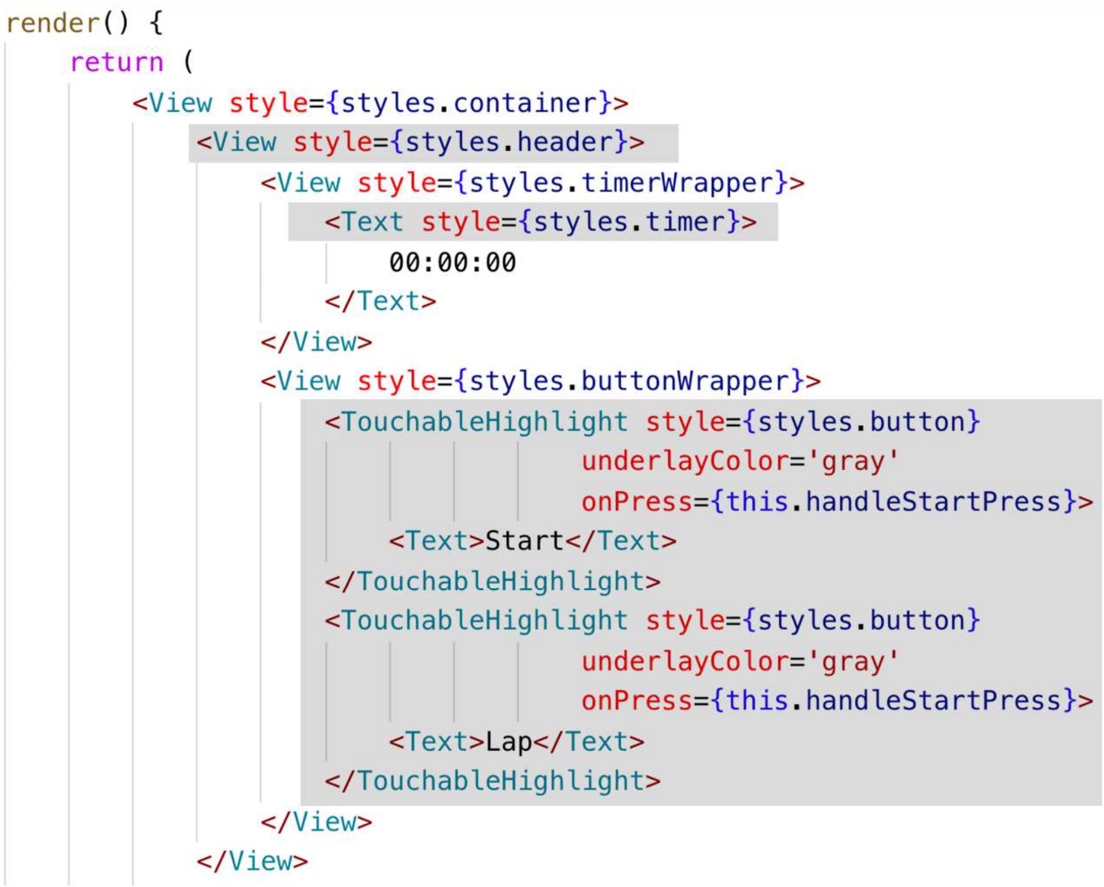
Lap #3 00:02.19

Lap #4 00:03.95

See The Difference

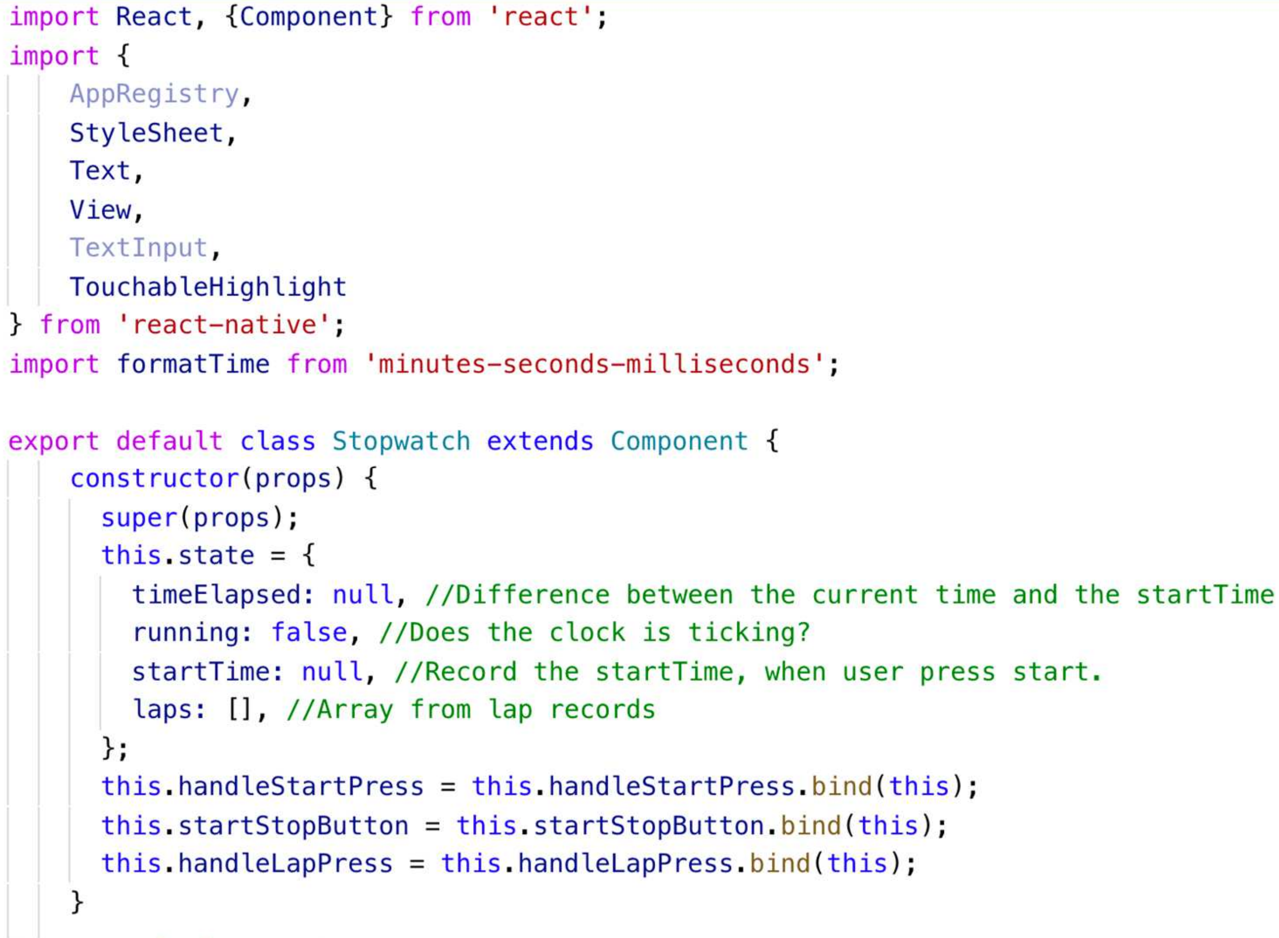


```
render() {
  return <View style={styles.container}>
    <View style={styles.header}>
      <View style={styles.timerWrapper}>
        <Text style={styles.timer}>
          {formatTime(this.state.timeElapsed)}
        </Text>
      </View>
      <View style={styles.buttonWrapper}>
        {this.lapButton()}
        {this.startStopButton()}
      </View>
    </View>
    <View style={styles.footer}>
      {this.laps()}
    </View>
  </View>
```



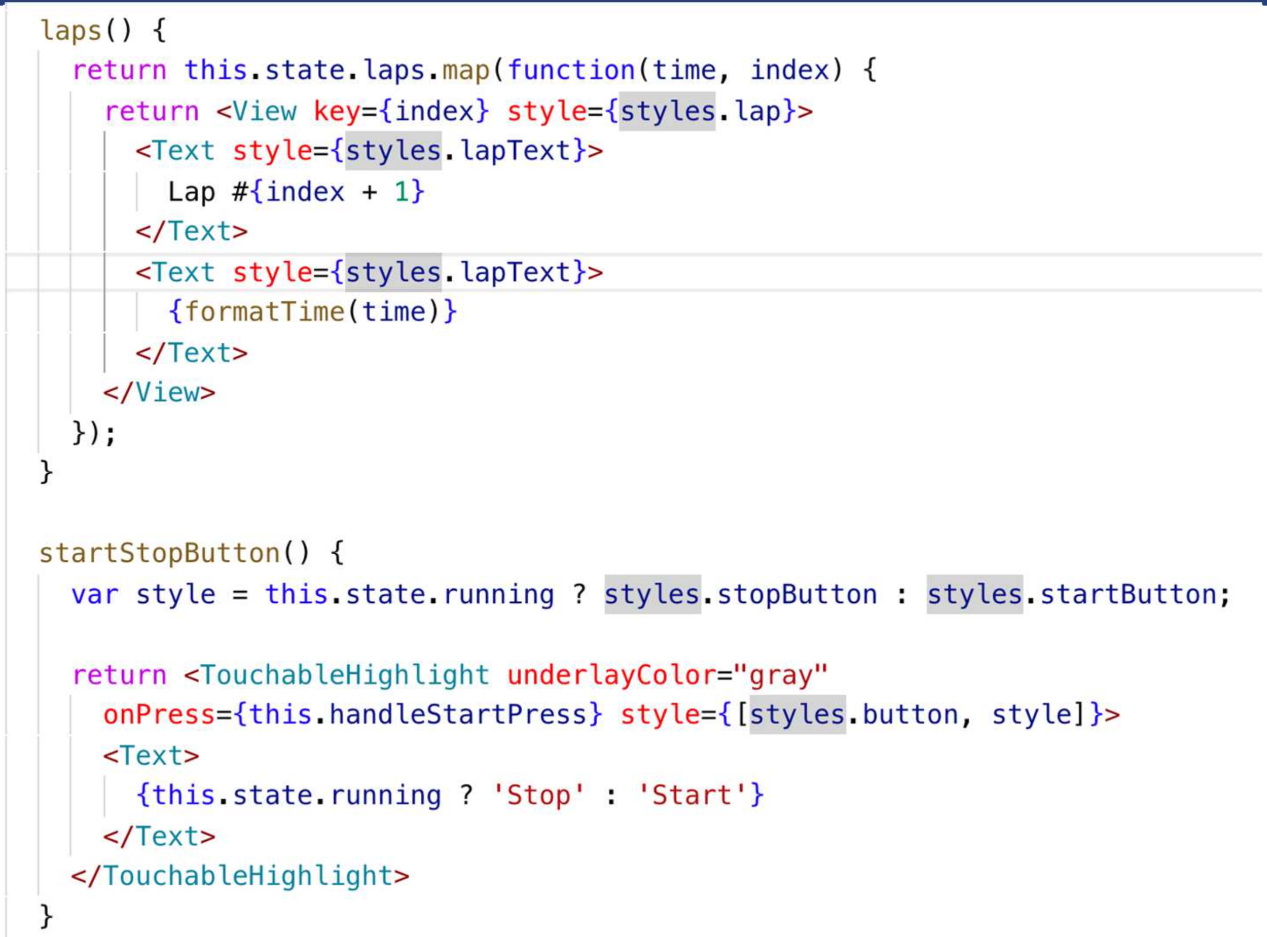
```
render() {
    return (
        <View style={styles.container}>
            <View style={styles.header}>
                <View style={styles.timerWrapper}>
                    <Text style={styles.timer}>
                        00:00:00
                    </Text>
                </View>
                <View style={styles.buttonWrapper}>
                    <TouchableHighlight style={styles.button}
                                underlayColor='gray'
                                onPress={this.handleStartPress}>
                        <Text>Start</Text>
                    </TouchableHighlight>
                    <TouchableHighlight style={styles.button}
                                underlayColor='gray'
                                onPress={this.handleStartPress}>
                        <Text>Lap</Text>
                    </TouchableHighlight>
                </View>
            </View>
```

Full Source Code #1



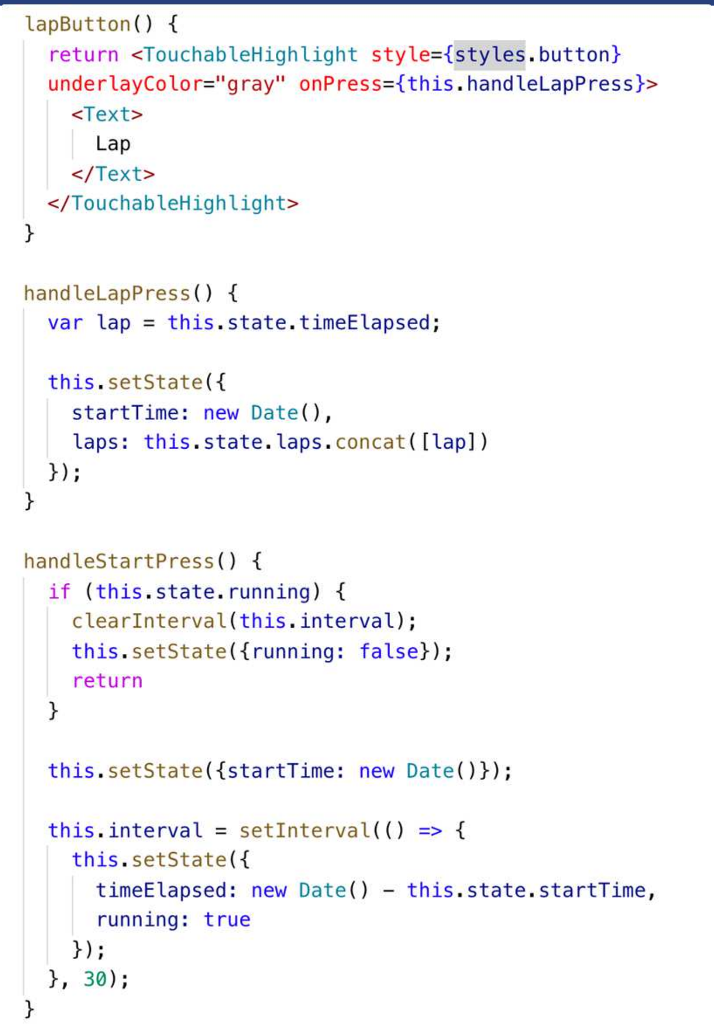
```
import React, {Component} from 'react';
import {
    AppRegistry,
    StyleSheet,
    Text,
    View,
    TextInput,
    TouchableHighlight
} from 'react-native';
import formatTime from 'minutes-seconds-milliseconds';

export default class Stopwatch extends Component {
    constructor(props) {
      super(props);
      this.state = {
        timeElapsed: null, //Difference between the current time and the startTime
        running: false, //Does the clock is ticking?
        startTime: null, //Record the startTime, when user press start.
        laps: [], //Array from lap records
      };
      this.handleStartPress = this.handleStartPress.bind(this);
      this.startStopButton = this.startStopButton.bind(this);
      this.handleLapPress = this.handleLapPress.bind(this);
    }
```

Full Source Code #2

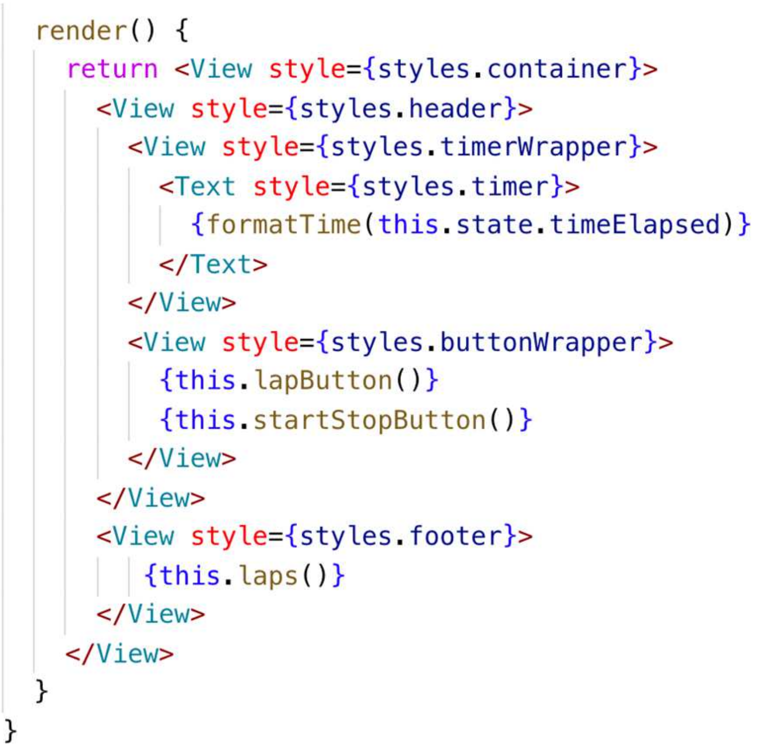


```
laps() {
  return this.state.laps.map(function(time, index) {
    return <View key={index} style={styles.lap}>
      <Text style={styles.lapText}>
        Lap #{index + 1}
      </Text>
      <Text style={styles.lapText}>
        {formatTime(time)}
      </Text>
    </View>
  });
}

startStopButton() {
  var style = this.state.running ? styles.stopButton : styles.startButton;

  return <TouchableHighlight underlayColor="gray"
    onPress={this.handleStartPress} style={[styles.button, style]}>
    <Text>
      {this.state.running ? 'Stop' : 'Start'}
    </Text>
  </TouchableHighlight>
}
```

Full Source Code 3

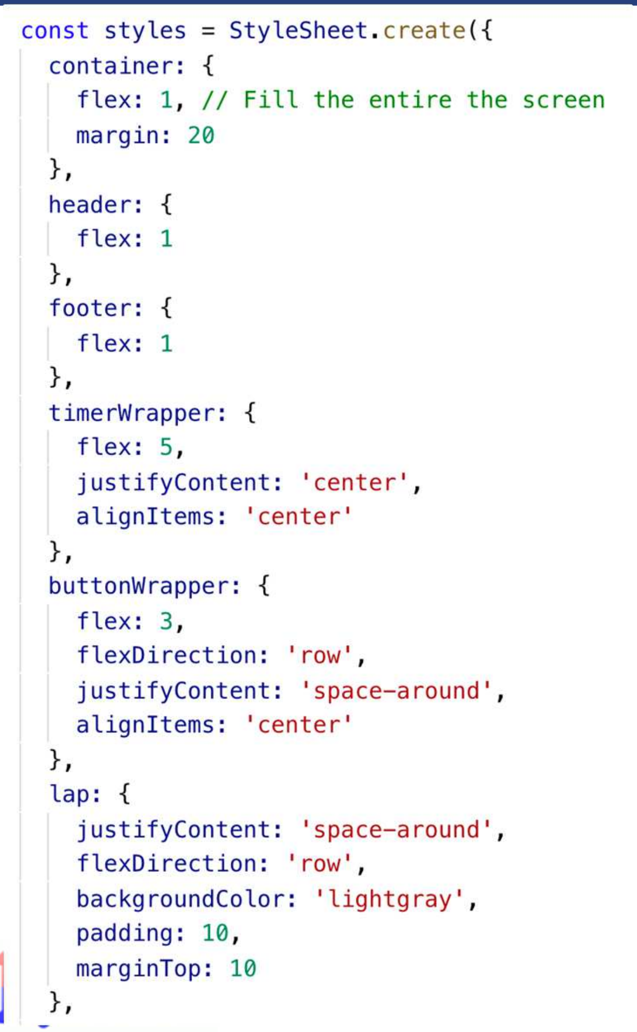


```
lapButton() {
  return <TouchableHighlight style={styles.button}
  underlayColor="gray" onPress={this.handleLapPress}>
    <Text>
      Lap
    </Text>
  </TouchableHighlight>
}

handleLapPress() {
  var lap = this.state.timeElapsed;

  this.setState({
    startTime: new Date(),
    laps: this.state.laps.concat([lap])
  });
}

handleStartPress() {
  if (this.state.running) {
    clearInterval(this.interval);
    this.setState({running: false});
    return
  }

  this.setState({startTime: new Date()});

  this.interval = setInterval(() => {
    this.setState({
      timeElapsed: new Date() - this.state.startTime,
      running: true
    });
  }, 30);
}
```

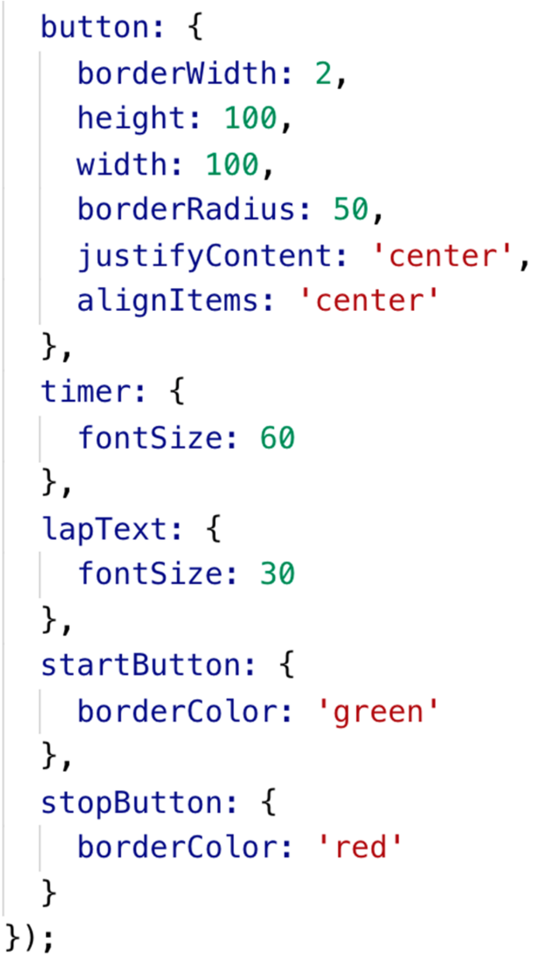


```
render() {
  return <View style={styles.container}>
    <View style={styles.header}>
      <View style={styles.timerWrapper}>
        <Text style={styles.timer}>
          {formatTime(this.state.timeElapsed)}
        </Text>
      </View>
      <View style={styles.buttonWrapper}>
        {this.lapButton()}
        {this.startStopButton()}
      </View>
    </View>
    <View style={styles.footer}>
      {this.laps()}
    </View>
  </View>
}
}
```

Full Source Code #4

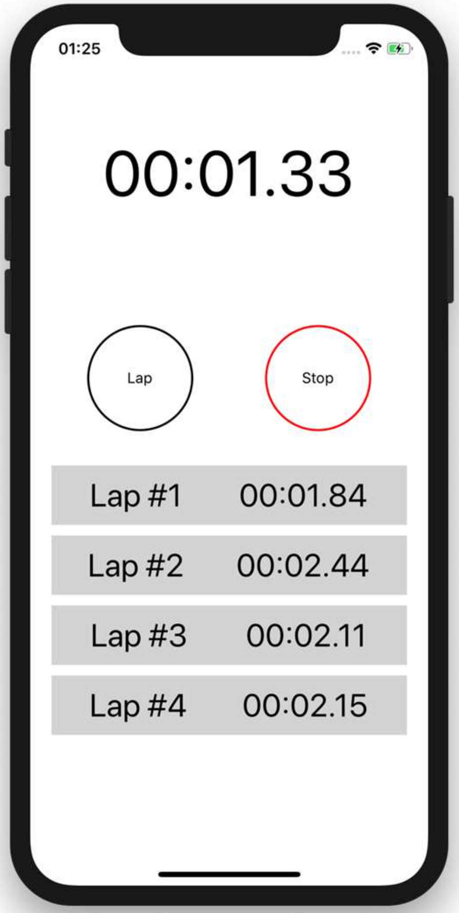


```
const styles = StyleSheet.create({
  container: {
    flex: 1, // Fill the entire the screen
    margin: 20
  },
  header: {
    flex: 1
  },
  footer: {
    flex: 1
  },
  timerWrapper: {
    flex: 5,
    justifyContent: 'center',
    alignItems: 'center'
  },
  buttonWrapper: {
    flex: 3,
    flexDirection: 'row',
    justifyContent: 'space-around',
    alignItems: 'center'
  },
  lap: {
    justifyContent: 'space-around',
    flexDirection: 'row',
    backgroundColor: 'lightgray',
    padding: 10,
    marginTop: 10
  },
```



```
  button: {
    borderWidth: 2,
    height: 100,
    width: 100,
    borderRadius: 50,
    justifyContent: 'center',
    alignItems: 'center'
  },
  timer: {
    fontSize: 60
  },
  lapText: {
    fontSize: 30
  },
  startButton: {
    borderColor: 'green'
  },
  stopButton: {
    borderColor: 'red'
  }
});
```

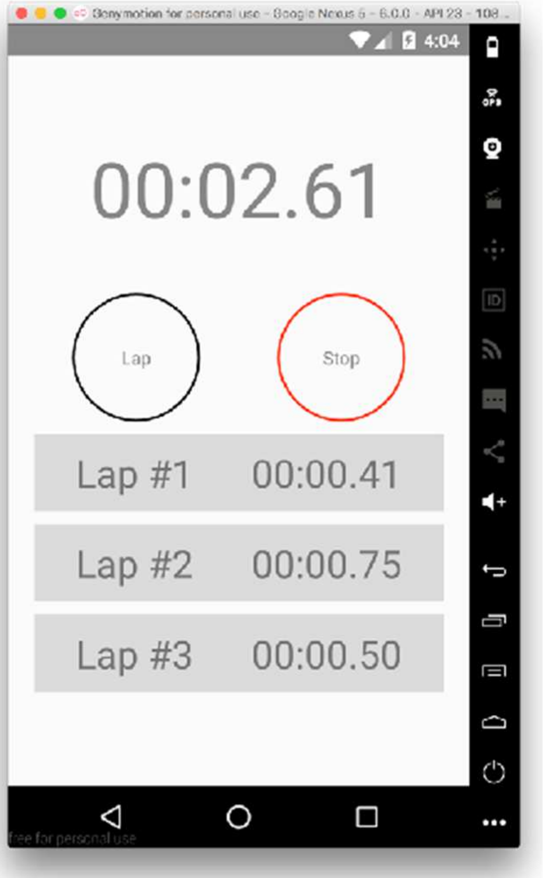
Result



00:01.33

Lap #1	00:01.84
Lap #2	00:02.44
Lap #3	00:02.11
Lap #4	00:02.15

iPhone X — 12.4


Genymotion for personal use - Google Nexus 5 - 6.0.0 - API 23 - 108 ...
4:04
Lap
Stop
free for personal use

00:02.61

Lap #1	00:00.41
Lap #2	00:00.75
Lap #3	00:00.50

Home Work

- How to reset the laps?
- Scrollable Laps?
- It is leaved as the homework for you
- See the iOS 10’s stopwatch app for idea

Exercise

- Design and program the basic calculator

Thank you!