REACT NATIVE

Component, Props, Network

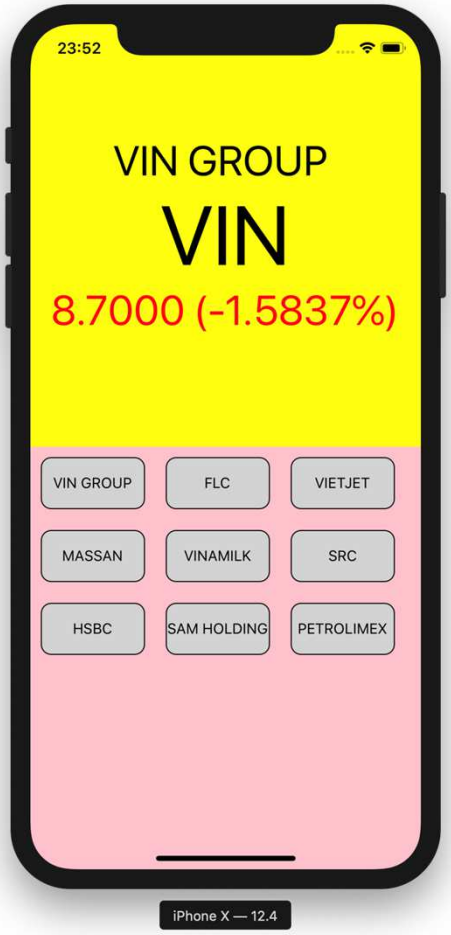
Making Stock App

- Draw UI on Paper
- Initial project using React-Native
- Making UI
- Making Styles
- Input Handling
- Fetch Data from Network

HIENLTH

Making UI

- Make a View have style.container fill the full screen
- Split them into two parts equally, header and footer (yellow and pink part)
- Yellow part, set everything on center
- Pink part, have flexDirection: ‘row’ and flexWrap: ‘wrap’



App.js

```
<View style={styles.container}>
    <View style={styles.header}>
        <Text style={styles.stockName}>
            VIN GROUP
            </Text>
        <Text style={styles.stockIndex}>
            8.7000
            </Text>
        <Text style={[styles.stockChange, style]}>
            8.7000 (-1.5837%)
            </Text>
    </View>
```


```
container:{
    flex:1
},
header:{
    flex:1,
    justifyContent:'center',
    alignItems:'center',
    backgroundColor: 'yellow'
},
stockName: {
  fontSize: 40
},
stockIndex: {
  fontSize: 80
},
stockChange: {
  fontSize: 40
},
```

App.js

```
<View style={styles.footer}>
    <TouchableOpacity style={styles.button}>
        <Text>VIN GROUP</Text>
    </TouchableOpacity>
    <TouchableOpacity style={styles.button}>
        <Text>FLC</Text>
    </TouchableOpacity>
    .....
    <TouchableOpacity style={styles.button}>
        <Text>SAM HOLDING</Text>
    </TouchableOpacity>
    <TouchableOpacity style={styles.button}>
        <Text>PETROLIMEX</Text>
    </TouchableOpacity>
</View>
```


```
footer:{
    flex:1,
    flexDirection: 'row',
    flexWrap: 'wrap',
    backgroundColor: 'pink'
},
button: {
    margin: 10,
    borderWidth: 1,
    width: 100,
    height: 50,
    borderRadius: 10,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: 'lightgray'
}
```

React's Component

- React is fundamentally designed for component-based programming
- Build encapsulated components that manage their own state, then compose them to make complex Uis application
- A component can have many children components
- Communication from parent to children components can be done via Props.

Make Repeat Elements As Components

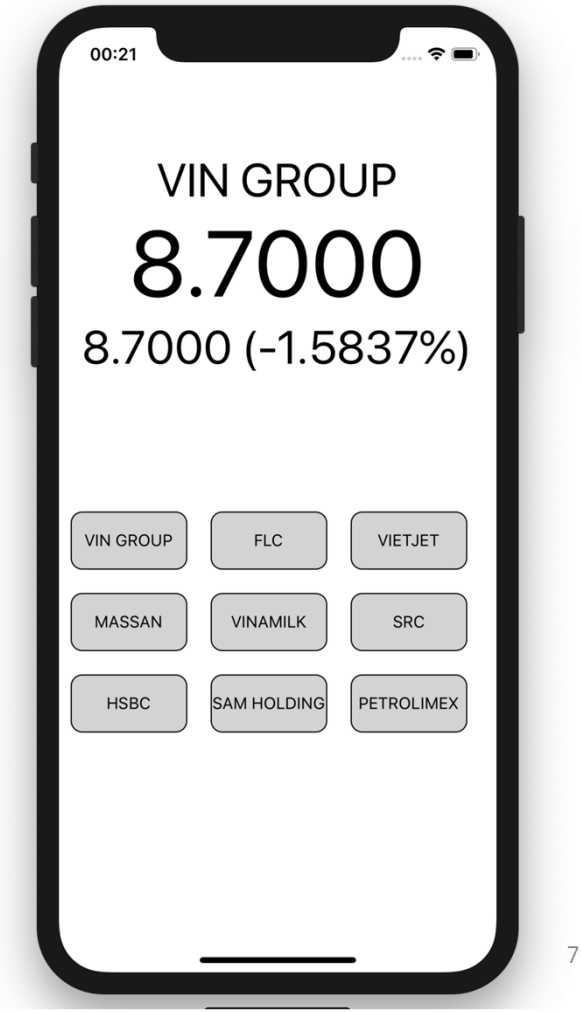
- We have 9 buttons for stock codes. (A button is highlighted on the left)
- Many duplicated elements should make it as a Component.
- Don’t Repeat Yourself! (DRY) principle.

```
<View style={styles.footer}>
    <TouchableOpacity style={styles.button}>
        <Text>VIN GROUP</Text>
    </TouchableOpacity>
    <TouchableOpacity style={styles.button}>
        <Text>FLC</Text>
    </TouchableOpacity>
    .....
    <TouchableOpacity style={styles.button}>
        <Text>SAM HOLDING</Text>
    </TouchableOpacity>
    <TouchableOpacity style={styles.button}>
        <Text>PETROLIMEX</Text>
    </TouchableOpacity>
</View>
```

HIENLTH

Hello StockButton!

- Create a new file named '**StockButton**.js' located in the same directory of the project
- StockButton is a component for represent a Stock Index button at the lower half of the app
- After we complete the StockButton component, we will import and use it as a part of the main Stock application component.



HIENLTH

StockButton.js

```
import React, { Component } from 'react';

import {
    StyleSheet,
    Text,
    TouchableOpacity
} from 'react-native';

export default class StockButton extends Component {
    render() {
        return(
            <TouchableOpacity style={styles.button}
                              onPress={()=>{
                                  this.props.onPress(
                                      this.props.name,
                                      this.props.code
                                  )
                              }}>
                <Text>{this.props.name}</Text>
            </TouchableOpacity>
        );
    }
}
```

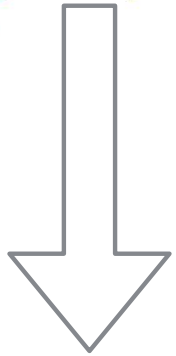
```
const styles = StyleSheet.create({
    button:{
        margin: 10,
        height: 50,
        width: 100,
        borderWidth: 1,
        borderRadius: 10,
        alignItems: 'center',
        justifyContent: 'center',
        backgroundColor: 'lightgray'
    }
});
```

Including StockButton Components in Main App

```
import {AppRegistry,
        StyleSheet,
        Text,
        View,
        TouchableOpacity} from 'react-native';
import StockButton from './StockButton.js';

export default class App extends Component {
    constructor(props){
        super(props);
```

```
<View style={styles.footer}>
    <TouchableOpacity style={styles.button}>
        <Text>VIN GROUP</Text>
    </TouchableOpacity>
    <TouchableOpacity style={styles.button}>
        <Text>FLC</Text>
    </TouchableOpacity>
```



```
<View style={styles.footer}>
    <StockButton name="VIN GROUP" VIN="" onPress={this.changeIndex}/>
    <StockButton name="FLC" code="FLC" onPress={this.changeIndex}/>
    <StockButton name="VIETJET" code="VJC" onPress={this.changeIndex}/>
    <StockButton name="MASSAN" code="MSN" onPress={this.changeIndex}/>
    <StockButton name="VINAMILK" code="VNM" onPress={this.changeIndex}/>
```

Adding ChangeIndex Method

```
export default class App extends Component {
    constructor(props){
        super(props);
        this.changeIndex = this.changeIndex.bind(this);
    }

    changeIndex(stockName, stockCode){
        console.log(stockName, stockCode);
    }
```

Push A Button & See Action in Remote Debugger

```
Running "lesson_4" with {"rootTag":151,"initialProps":{}}   setUpDeveloperTools.js:73
FLC FLC                                                     setUpDeveloperTools.js:73
VIETJET VJC                                                 setUpDeveloperTools.js:73
VINAMILK VNM                                                setUpDeveloperTools.js:73
SRC SRC                                                     setUpDeveloperTools.js:73
>
```

Props

- Communication from parent to child components
- It can be done by using attributes values, and/or callback functions
- Callback functions allows the child components to communicate back to the parents

```
changeIndex(stockName, stockCode){
    API(stockCode).then((data) => {
        console.log(data);
        this.setState({...data, stockName});
    });
}
```

A Callback Function

```
<StockButton name="VIN GROUP" code="VIN" onPress={this.changeIndex}/>
```

Attribute Value Attribute Value A Callback Function

App Component

StockButton Component

```
changeIndex(stockName, stockCode){
    API(stockCode).then((data) => {
        console.log(data);
        this.setState({...data, stockName});
    });
}
```

```
<StockButton name="VIN GROUP" code="VIN" onPress={this.changeIndex}/>
```

this.prop

```
render() {
    console.log(this.props.name);    // VIN GROUP
    console.log(this.props.code);    // VIN
    console.log(this.props.onPress);// [Function]
    return(
```

```
<TouchableOpacity style={styles.button}
                  onPress={()=>{
                      this.props.onPress(
                          this.props.name,
                          this.props.code
                      )
                  }}>
    <Text>{this.props.name}</Text>
</TouchableOpacity>
```

Index Component

App Component

```
changeIndex(stockName, stockCode){
    API(stockCode).then((data) => {
        console.log(data);
        this.setState({...data, stockName});
    });
}

<StockButton name="VIN GROUP" code="VIN" onPress={this.changeIndex}/>
```

A Callback Function

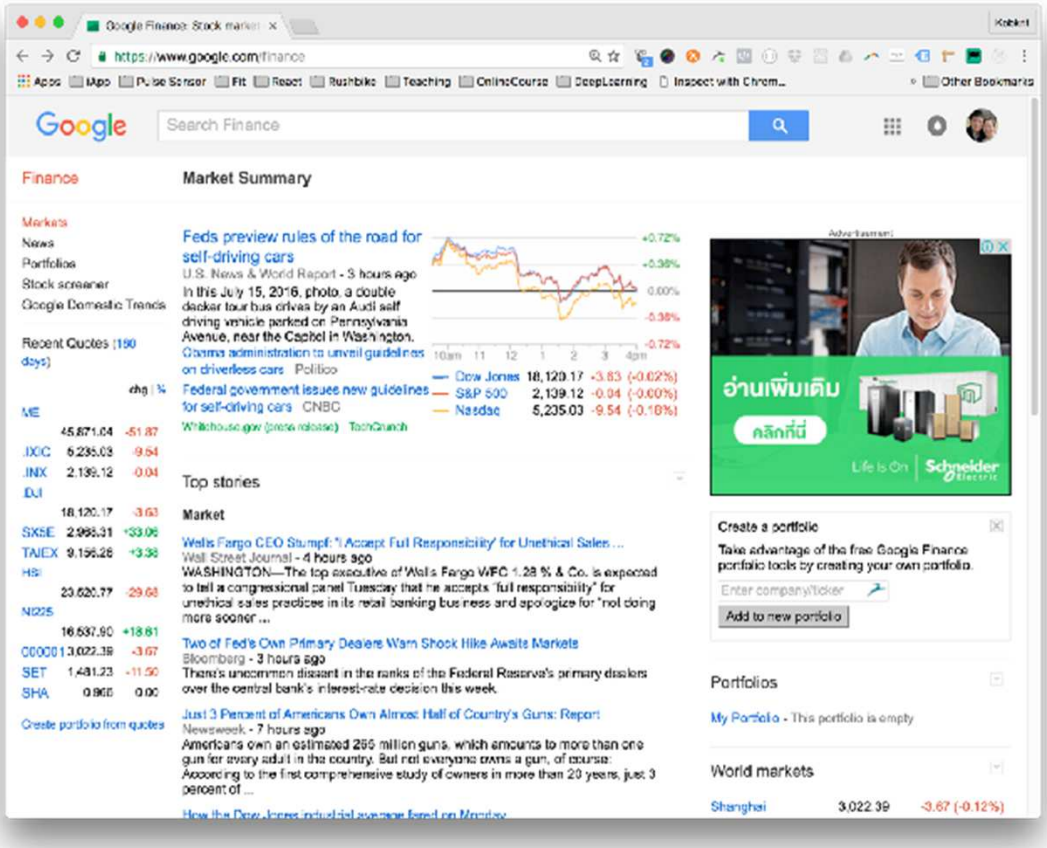
When User Press a Button

```
render() {
    console.log(this.props.name);    // VIN GROUP
    console.log(this.props.code);    // VIN
    console.log(this.props.onPress);// [Function]
    return(
```

```
<TouchableOpacity style={styles.button}
                  onPress={()=>{
                      this.props.onPress(
                          this.props.name,
                          this.props.code
                      )
                  }}>
    <Text>{this.props.name}</Text>
</TouchableOpacity>
```

StockButton Component

Getting Stock Information



Google Finance https://finance.google.com/finance

Getting Index Codes

World markets

Shanghai	3,022.51	-3.54 (-0.12%)
Nikkei 225	16,537.90	+18.61 (0.11%)
Hang Seng Index	23,515.09	-35.36 (-0.15%)
TSEC	9,157.34	+4.46 (0.05%)
FTSE 100	6,813.55	+103.27 (1.54%)
EURO STOXX 50	2,968.31	+33.06 (1.13%)
CAC 40	4,394.19	+61.74 (1.43%)
S&P TSX	14,496.23	+45.39 (0.31%)
S&P/ASX 200	5,283.20	-11.60 (-0.22%)
BSE Sensex	28,634.50	+35.47 (0.12%)
TA25	1,448.67	+1.23 (0.08%)
SMI	8,195.71	+65.27 (0.80%)
ATX	2,350.10	+29.40 (1.27%)
IBOVESPA	57,350.38	+270.62 (0.47%)
SET	1,478.63	-14.10 (-0.94%)
BIST100	77,670.75	+1,650.46 (2.17%)
IBEX	8,715.50	+82.10 (0.95%)
WIG	47,405.08	+238.41 (0.51%)
TASI	5,935.96	-124.11 (-2.05%)
MERVAL	14,928.77	+250.70 (1.71%)
IPC	45,871.04	-51.87 (-0.11%)
IDX Composite	5,305.07	-16.77 (-0.32%)

Getting JSON Information Of a market index

- ALPHA VANTAGE have internal API for getting finance information in JSON format
- https://www.alphavantage.co/query?function=GLOBAL_QUOTE&apikey=CNDJO9WP98PVJP38&symbol=VIN


```
{
    "Global Quote": {
        "01. symbol": "VIN",
        "02. open": "8.7000",
        "03. high": "9.0000",
        "04. low": "8.6000",
        "05. price": "8.7000",
        "06. volume": "24832",
        "07. latest trading day": "2018-01-30",
        "08. previous close": "8.8400",
        "09. change": "-0.1400",
        "10. change percent": "-1.5837%"
    }
}
```

Hint

- Register an account to get KEY for calling API https://www.alphavantage.co
- Test API for 01 specific code, for example: IBM https://www.alphavantage.co/query?function=GLOBAL_QUOTE&apikey=KRS76I47DA16G9W7&symbol=IBM
- The JSON result return:


```
{
    "Global Quote": {
        "01. symbol": "IBM",
        "02. open": "127.3800",
        "03. high": "128.9300",
        "04. low": "126.3700",
        "05. price": "128.7600",
        "06. volume": "7400216",
        "07. latest trading day": "2022-03-18",
        "08. previous close": "127.9600",
        "09. change": "0.8000",
        "10. change percent": "0.6252%"
    }
}
```

JSON Explain


```
{
        "Global Quote": {
            "01. symbol": "VIN",
            "02. open": "8.7000",
            "03. high": "9.0000",
            "04. low": "8.6000",
            "05. price": "8.7000",
            "06. volume": "24832",
            "07. latest trading day": "2018-01-30",
            "08. previous close": "8.8400",
            "09. change": "-0.1400",
            "10. change percent": "-1.5837%"
        }
    }
```

API Module

- Create a new file named ‘api.js’ located in the same directory of the project
- Export as the default function that getting Stock information from the Alpha Vantage
- Main app (App.js) will imported that API module and using it in the changeStock method.

api.js

```
let rootURL = 'https://www.alphavantage.co/query?function=GLOBAL_QUOTE&apikey=CNDJ09WP98PVJP38&symbol=';

export default function(code){
    let url = `${rootURL}${code}`;
    return fetch(url).then(function(response){
        return response.text();
    }).then(function(text){
        let rawJSONString = text.replace("//", "");
        let json = JSON.parse(rawJSONString);
        let data = json["Global Quote"];
        return {
            stockIndex: data["01. symbol"],
            stockChangeRaw: data["02. open"],
            stockChangePercent: data["10. change percent"]
        };
    });
}
```

Fetch API

- The Fetch API provides an interface for fetching resources (including across the network)
- It is a living standard in Web Hypertext Application Technology Working Group (WHATWG)
- Syntax:

```
fetch(url)
.then(function(response) {
    response.blob();          // Resolve response as Blob
    response.json();          // Resolve response as JSON
    response.response.text(); // Resolve response as Text
}).then(function(data) {
    //Using data
});
```

HIENLTH

https://reactnative.dev/docs/network

.then Promise

- .then promise is a kind of advance technique of chaining callback function
- .then promise will be invoked when the earlier operation is completed

```
fetch(url)                          // (1) Fetching from URL
.then(function(response) {          // (2) Wait until fetching is finished,
    response.blob();                // then convert response data to text format
    response.json();
    response.response.text();
}).then(function(data) {            // (3) Wait until the data conversion
    //Using data                    // is finished, then, using the data.
});
```

.catch Promise

- In cases of fetching errors, such as, Internet connection is down, the URL is no longer exist, etc. How can we resolve it?
- .catch promise can be added at the end of any promise to capture any errors in the promise. The code interpreter will jump from the point of the error occur to the functions in .catch Promise.

```
fetch('flowers.jpg')
    .then(function(response) {
        if(response.ok) {
            response.blob().then(function(myBlob) {
                var objectURL = URL.createObjectURL(myBlob);
                myImage.src = objectURL;
            });
        } else {
            console.log('Network response was not ok.');
        }
})
.catch(function(error) {
    console.log('There has been a problem with your fetch operation ' + error.message);
});
```

api.js

```
return fetch(url).then(function(response){
    return response.text();
}).then(function(text){
    let rawJSONString = text.replace("//", "");
    let json = JSON.parse(rawJSONString);
    let data = json["Global Quote"];
    return {
        stockIndex: data["01. symbol"],
        stockChangeRaw: data["02. open"],
        stockChangePercent: data["10. change percent"]
    };
});
```


```
{
    "Global Quote": {
        "01. symbol": "VIN",
        "02. open": "8.7000",
        "03. high": "9.0000",
        "04. low": "8.6000",
        "05. price": "8.7000",
        "06. volume": "24832",
        "07. latest trading day": "2018-01-30",
        "08. previous close": "8.8400",
        "09. change": "-0.1400",
        "10. change percent": "-1.5837%"
    }
}
```

Output the value

- Set up state for displaying the value
- Set up initial state

```
export default class App extends Component {
    constructor(props){
        super(props);
        this.state = {
          stockName: 'VIN',
          stockIndex:'0.00',
          stockChangeRaw:'+0.00',
          stockChangePercent:'+0.00'
        };
        this.changeIndex = this.changeIndex.bind(this);
    }
```

Update value

```
<View style={styles.container}>
    <View style={styles.header}>
        <Text style={styles.stockName}>
            {this.state.stockName}
        </Text>
        <Text style={styles.stockIndex}>
            {this.state.stockIndex}
        </Text>
        <Text style={[styles.stockChange, style]}>
            {this.state.stockChangeRaw} ({this.state.stockChangePercent})
        </Text>
    </View>
```

Update State

```
changeIndex(stockName, stockCode){
    API(stockCode).then((data) => {
        console.log(data);
        this.setState({...data, stockName});
    });
}
```


01:22
VIN
0.00
+0.00 (+0.00)
VIN GROUP
FLC
VIETJET
MASSAN
VINAMILK
SRC
HSBC
SAM HOLDING
PETROLIMEX
iPhone X — 12.4
01:23
FLC
FLC
22.3800 (1.0314%)
VIN GROUP
FLC
VIETJET
MASSAN
VINAMILK
SRC
HSBC
SAM HOLDING
PETROLIMEX
iPhone X — 12.4

Making Initial Loading

Add this.changeIndex at the constructor

```
export default class App extends Component {
    constructor(props){
        super(props);
        this.state = {
          stockName: 'VIN',
          stockIndex:'0.00',
          stockChangeRaw:'+0.00',
          stockChangePercent:'+0.00'
        };
        this.changeIndex = this.changeIndex.bind(this);
        this.changeIndex('VIN GROUP ', 'VIN');
    }
```

Adding Black/ Red/ Green Text Color

```
render(){
    let style = styles.red;
    if (this.state.stockChangeRaw[0] == '+'){
        style = styles.green;
    }
```

```
red:{
    color: 'red'
},
green:{
    color: 'green'
}
```

Final App


01:28
VIN GROUP
VIN
8.7000 (-1.5837%)
VIN GROUP
FLC
VIETJET
MASSAN
VINAMILK
SRC
HSBC
SAM HOLDING
PETROLIMEX
iPhone X — 12.4

Thank you!