Introduction to React.js

Mar 16th 2019 NUS Hacker School

Shi Tianyuan

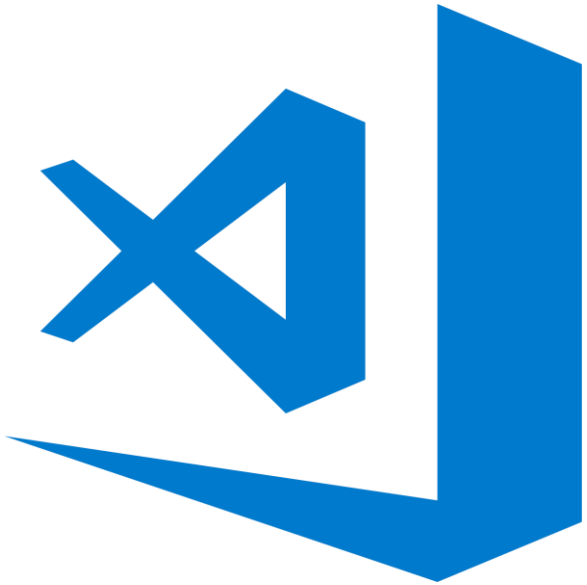
What you need before getting started

Node.js LTS or newer version

(Our Development Environment)

Visual Studio Code

(A fantastic text editor)



What is React.js

“

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.

”

It does the same thing as your normal HTML + CSS + JavaScript, but in a different and more efficient manner.

What is React.js

- Component-based library for building frontend UI
- JavaScript (JS)-based
- Each page is powered by many components
- Components can contain other components

Web Applications Written in React

- Outlook Web Mail
- Instagram
- Facebook
- WhatsApp Web

Why do we use React?

- Same reason why there's a need for frameworks in backend development
- For large-scale application, frontend development is non-trival
- Need to have a more systematic approach of doing development

Why do we use React?

- **TRADITIONAL JAVASCRIPT**
- State changes -> Define and make UI changes yourself (manual)

1. Listen for changes
2. Add a new todo item
3. Append new item to item list
4. Keep track of which new item was added
5. Use jQuery to update multiple parts of the UI
6. You need to **keep track** of state

Why do we use React?

REACT.JS

State changes -> UI re-renders based on new state (auto)

Benefits of React

- Helps to manage the DOM automatically
- Fast - Virtual DOM
- Large community
- Makes it easy to create Single Page Application

Overview

- Recap JavaScript (ES6) Knowledge
- React Development Basics
  - JSX
  - Component
  - State
  - Component Lifecycle
  - Props
- Create a TODO App on your own! (starter code given here)
  - Finished project effect

Recap

ES6 variable keyword, arrow function and class

Recap ES6 – let

The let statement declares a block scope local variable, optionally initializing it to a value.

```
let x = 1;

if (x === 1) {
let x = 2;

console.log(x);
// expected output: 2
}

console.log(x);
// expected output: 1
```

Recap ES6 – const

Constants are block-scoped, much like variables defined using the let statement. The value of a constant cannot change through reassignment, and it can't be redeclared.

```
const number = 42;

try {
number = 99;
} catch(err) {
console.log(err);
// expected output: TypeError: invalid assignment to const 'number'
// Note - error messages will vary depending on browser
}

console.log(number);
// expected output: 42
```

Recap ES6 – arrow function

Arrow functions are a different way of creating functions in JavaScript. Besides a shorter syntax, they offer advantages when it comes to keeping the scope of the this keyword

```
var elements = [
'Hydrogen',
'Helium',
'Lithium',
'Beryllium'
];
```

```
elements.map(function(element) {
    return element.length;
}); // this statement returns the array: [8, 6, 7, 9]

// The regular function above can be written as the arrow function below
elements.map((element) => {
    return element.length;
}); // [8, 6, 7, 9]

// When there is only one parameter, we can remove the surrounding
parenthesies:
elements.map(element => {
    return element.length;
}); // [8, 6, 7, 9]
```

Recap ES6 – Class

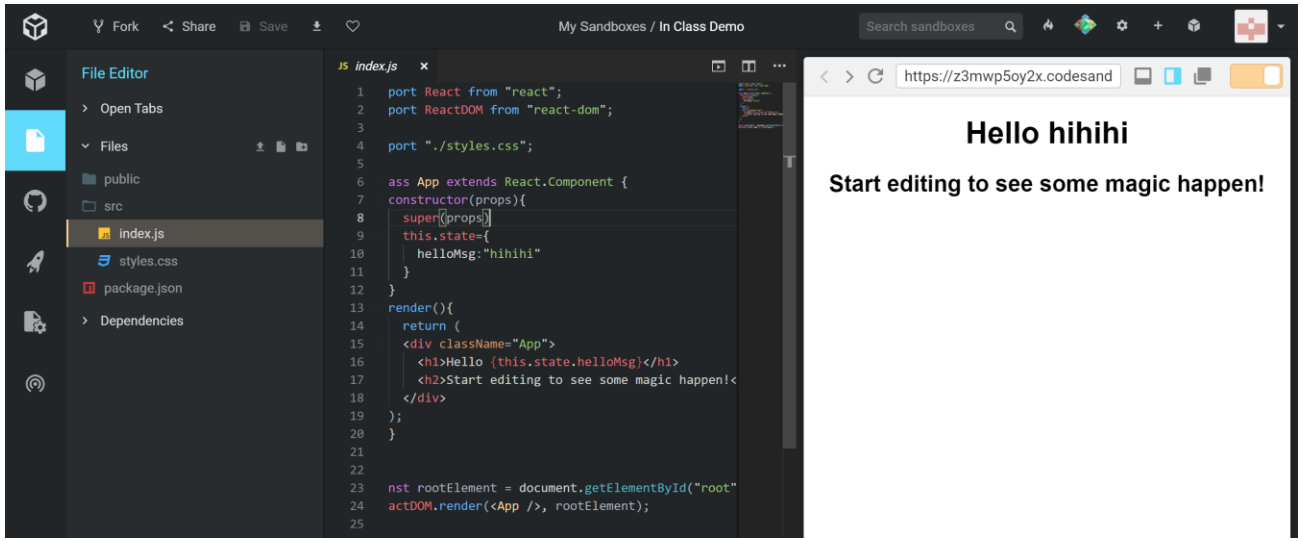
JavaScript Class is similar to Java Class, take note of the pattern, we are going to use this later

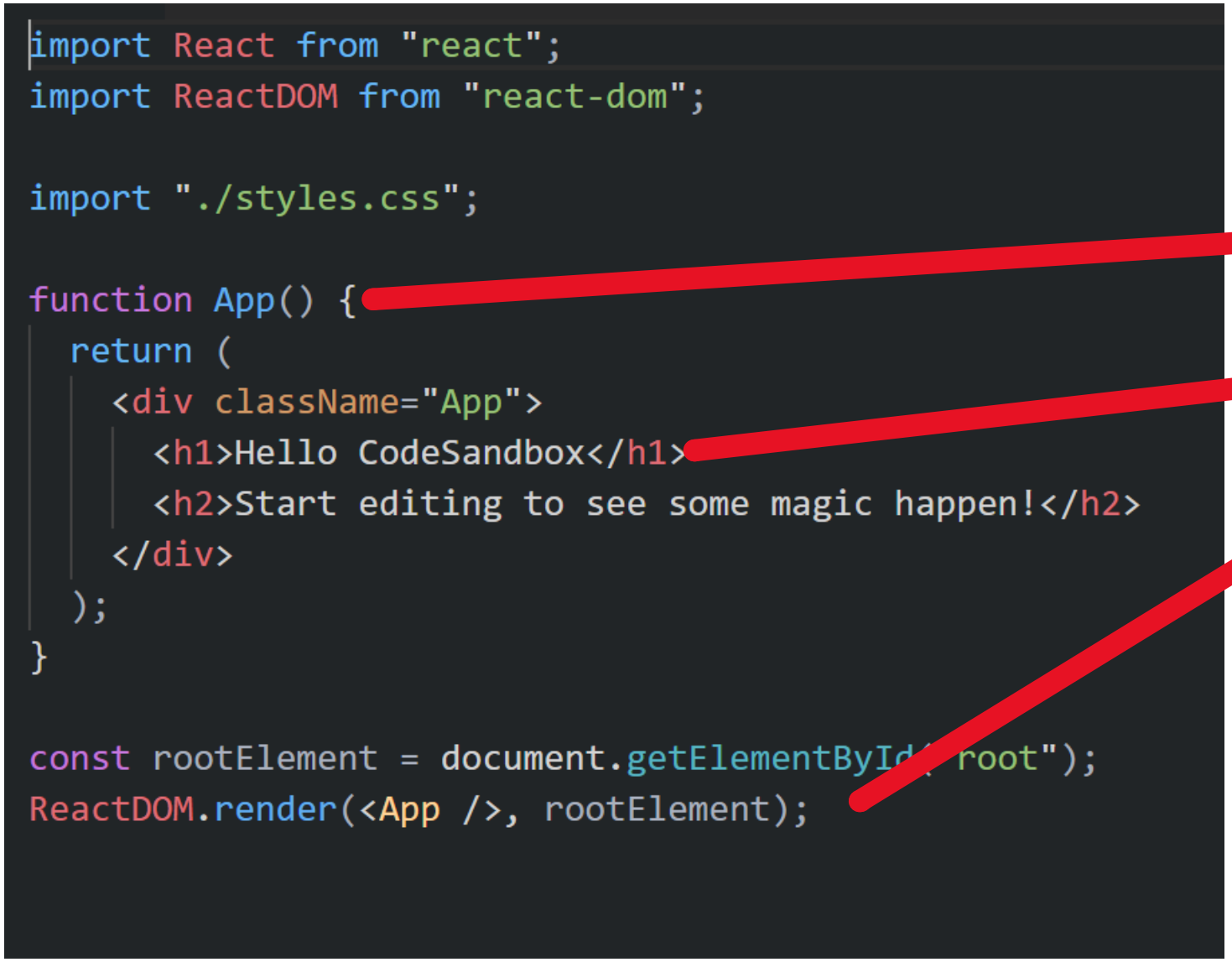
```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
// Getter
    get area() {
        return this.calcArea();
    }
// Method
    calcArea() {
        return this.height * this.width;
    }
}
```

Start with React

Online Editor

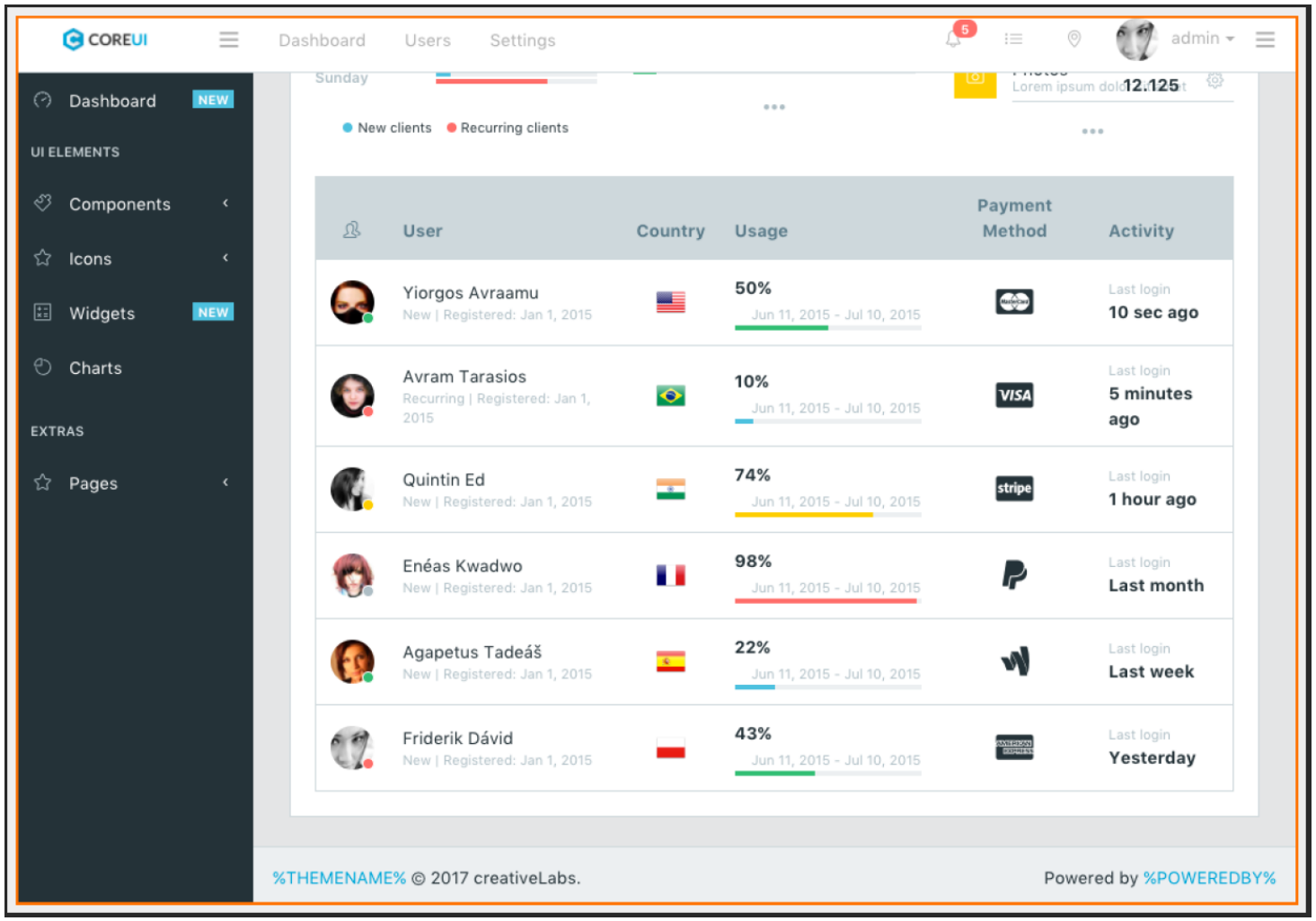
- Google codesandbox.io
- Login with your preferred account
- Create a React Sandbox
- We are good to go now!





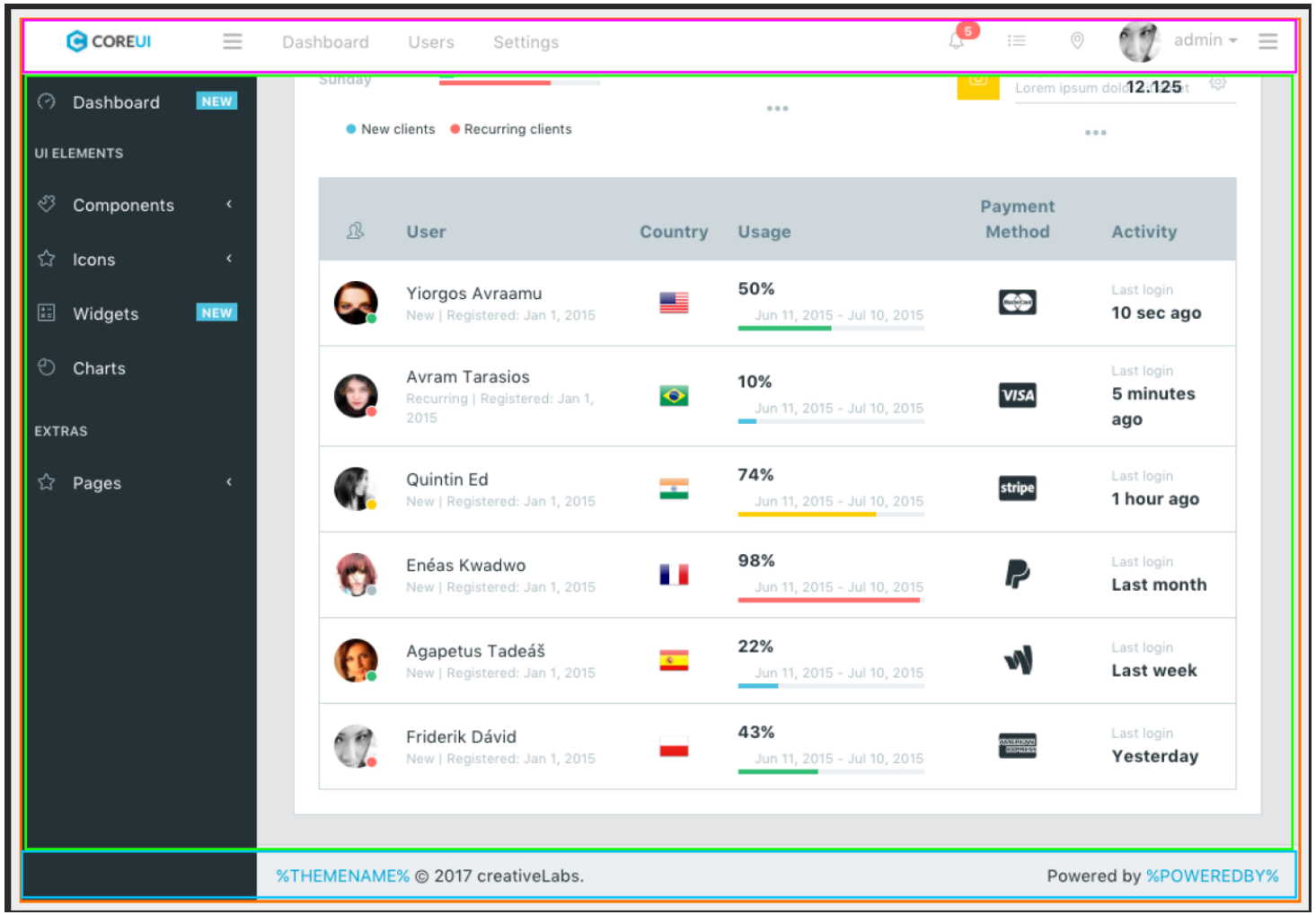
Explore the starter code

- Stateless Function Component
- JSX (it looks like HTML but it is not)
- How your components are rendered to your web browser



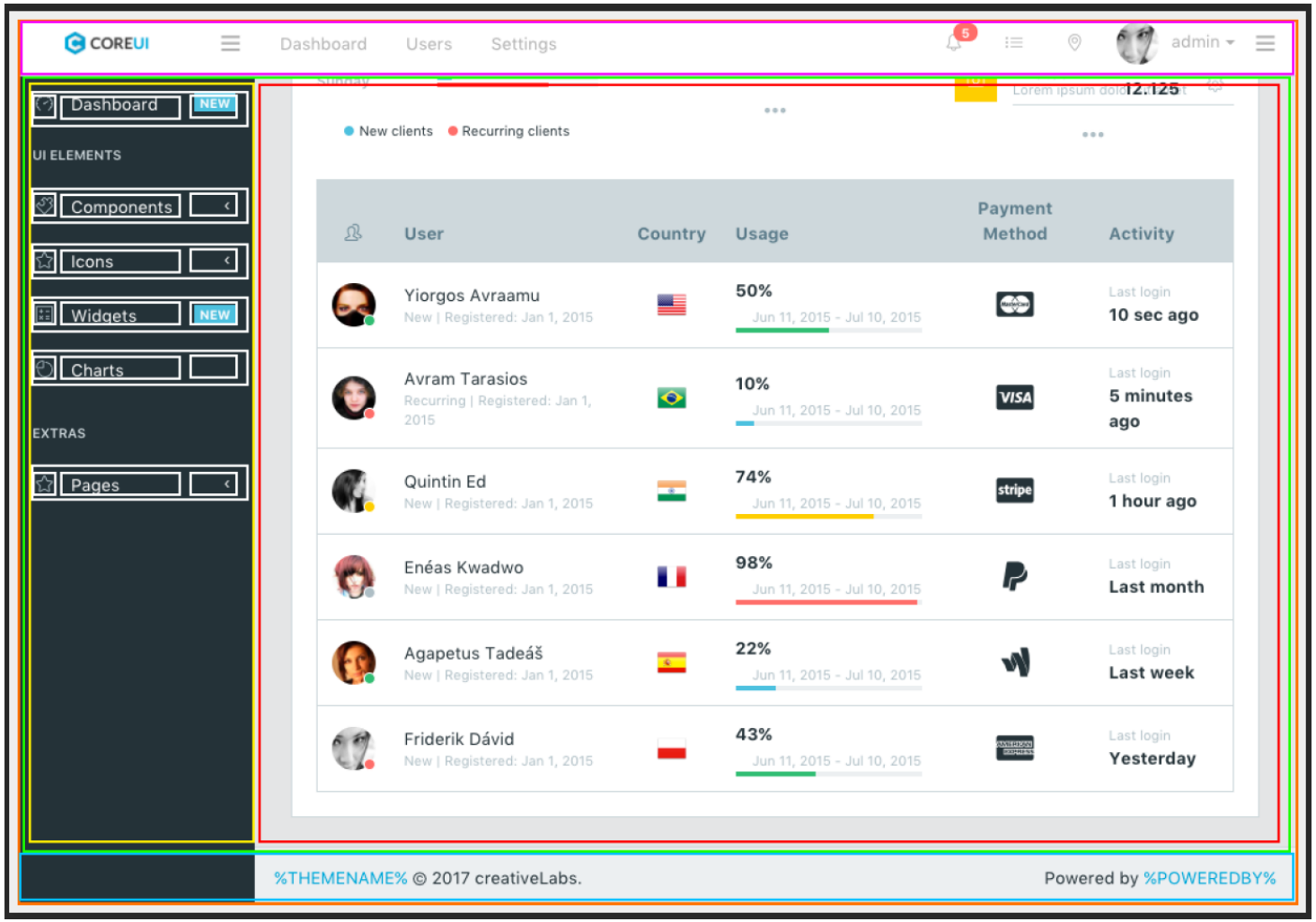
Components I

- Entire page consists of a component



Components II

- This component contains other components: Header, Body, Footer



Components III

- The SideBar can contain a MenuBar, which consists of MenuItems, etc

React Components

- React component extends the HTML vocabulary
- Coding style involve writing HTML inside the JS codes
- 3 ways to define components
  - React.createClass()
  - ES6 Component
  - Stateless function
- We will talk about state later
- Syntax Example on the next page

React.createClass

```
const MenuComponent = React.createClass({
render() {
    return (
        <div>
        <a href="http://www.google.com">Google</a><br />
        <a href="http://www.yahoo.com">Yahoo</a><br />
        <a href="http://www.facebook.com">Facebook</a><br />
        </div>
    );
}
});
```

ES6 Class Component

```
class MenuComponent extends React.Component {
render() {
    return (
        <div>
        <a href="http://www.google.com">Google</a><br />
        <a href="http://www.yahoo.com">Yahoo</a><br />
        <a href="http://www.facebook.com">Facebook</a><br
        />
        </div>
    );
}
}
```

Stateless Functions

```
const MenuComponent = () => {

return (
    <div>
    <a href="http://www.google.com">Google</a><br />
    <a href="http://www.yahoo.com">Yahoo</a><br />
    <a href="http://www.facebook.com">Facebook</a><br />
    </div>
);

}
```

```
render() {
  return (
    <div className="App">
      <h1>{this.state.helloMsg}</h1>
      {this.state.msgVisible && (
        <h2>Start editing to see some magic happen!</h2>
      )}

      <button
        onClick={() => {
          this.setState({ msgVisible: !this.state.msgVisible });
        }}
      >
        click me
      </button>
    </div>
  );
}
```

JSX

- Insert Dynamic Value
- Conditional Rendering
- Example

Props

- One of the main benefits of components is reusability
- Values can be supplied to components as props

```
<GreetingComponent first="John" last="Doe"/>
```

Hello John Doe

props: **{“first":"John", "last":"Doe"}**

```
class GreetingComponent extends React.Component{
    render(){
    return ( <p>Hello {this.props.first}
    {this.props.last}</p> );
    }
}
```

Props Passing Values

- Use { } in order to pass values as props when they are not string type
- e.g. numeric values and objects have to be enclosed with { }

```
<MyComponent
field1="some_string"
field2={10}
field3={["item1", "item2"]}
field4={{name:"John", age:10}}/>
```

State

- Props are often used for initial rendering but often the UI will change over time (e.g. due to events)
- Props is fixed (and supplied by the caller) while state will change over time
- See Example
- Notice how the tick function is triggered (component life cycle)

Summary

- This workshop has covered the basics of Web Development with React.js
  - JSX and Components Syntax
  - State and Props
- What has been taught today is just the basic syntax and definitions, but should be enough to get you started.

Next Steps

- React-Router for front-end routing
- Redux for state management
- React Native for Mobile App Development

You can start to build your first TODO App now!

starter code given here

Finished project effect

You may refer to the official tutorial in case you are not clear about anything.

References

- https://javascript101.netlify.com
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
- https://www.youtube.com/watch?v=3qk6yQWKVoQ
- Facebook React Tutorial
- Dr. Lek Hsiang Hui's IS3106 Notes (special thanks prof!)