ReactJS Technical Training

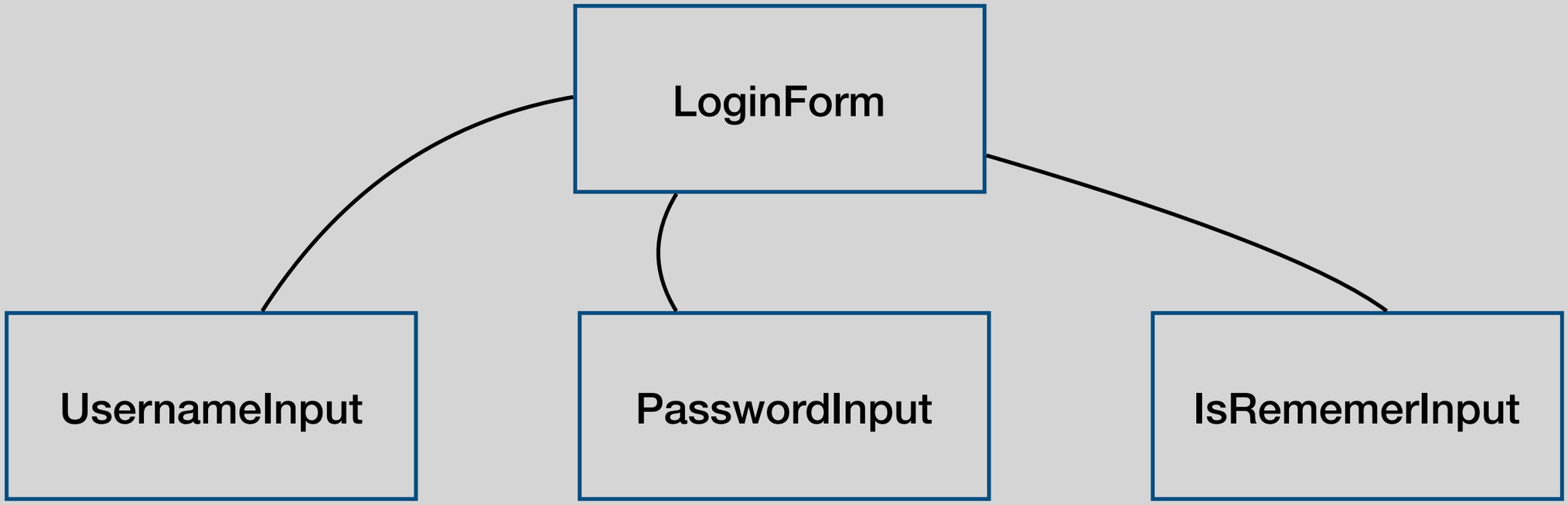
Justin Pham

- What is ReactJS?
- Why do we do use ReactJS?
- How do we use ReactJS?
- How ReactJS works in Mars project?

What is ReactJS?

- Again! Another JS library
- It’s internally developed by Facebook Developers and later become Open-source
- Only for building user interfaces. The view layer of the web | mobile application.
- .NET instead of server-rendering, React will take care of the view layer.
- In large-scale and well-budgeted project, architect designs seperate concern of layers. It means the back-end and front-end is decoupling. Back-end is API and Frontend is using React

- At the heart of React application are **components ??.**
    - Components are self-contained module that renders some output.
    - Component are composable **,** means you can include one or more components in it output
    - Component communicates to others by **props** and **state**


LoginForm
UsernameInput
PasswordInput
IsRememerInput

How does it work?

- Unlike other JS framework, React doesn’t operate directly on Document Object Model (DOM) but it’s on **Virtual DOM.** Sounds scary? It’s just internal JSON format file includes all config and setting.

- ReactJS only updates the actual DOM when **Virtual DOM** is updated

- For instance, KnockoutJS manipulate 2-way data-binding through document, every time when data updates, it’s rendered the entire DOM.

```
<html>
<head>
  <meta charset="utf-8">
  <title>Hello world</title>
  <!-- Script tags including React -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.3.1/react.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.3.1/react-dom.min.js"></script>
  <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
  <link href="https://gist.githubusercontent.com/auser/2bc34b9abf07f34f602dccd6ca855df1/raw/
  40c5e7c8cad4c6920fed940fc31cbb63abd94c29/timeline.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <div id="app"></div>
  <script type="text/babel">
    ReactDOM.render(
      <h1>Hello world</h1>,
      document.querySelector('#app')
    );
  </script>
</body>
</html>
```

ES5 | ES6 | JSX

- ES5 - a 5th update of JavaScript - finalised 2009 - support on all browsers

    - Var LoginForm = React.createClass()

- ES6 - a 6th update of JavaScript - finalised 2015 - more user-friendly and OOP language alike. Support on major browsers except IE (12% market share. who cares!)

    - Class LoginForm extends React.Component({})

- JSX - Javascript Extension - basically allow us to write HTML alike inside Javascript :)

Babel

- For those who care about the 12% marketshare. Babel is a library for transpiling ES6 to ES5

Now lets build the first component

```
class HelloWorld extends React.Component {
  render() {
    return (
      <h1 className='large'>Hello World</h1>
    );
  }
}
```

The JSX is translated to following JS at run time

```
class HelloWorld extends React.Component {
  render() {
    return (
      React.createElement(
        'h1',
        {className: 'large'},
        'Hello World'
      )
    );
  }
}
```

Complex Component

- Now you already built the first simple component
- All simple components works together will become complex components :)
- Container component & Parent | Child component
    - Container component: a wrapper of the application | page
    - Child component: when a component is nested inside another component, it’s called *child component*
    - Parent component: the component uses child component is then called parent component (of course)

```
class Header extends React.Component {
  render() {
    return (
      <div className="header">
        <div className="fa fa-more"></div>

        <span className="title">Timeline</span>

        <input
          type="text"
          className="searchInput"
          placeholder="Search ..." />

        <div className="fa fa-search searchIcon"></div>
      </div>
    )
  }
}
```

```
class Content extends React.Component {
  render() {
    return (
      <div className="content">
        <div className="line"></div>
        {/* Timeline item */}
        <div className="item">
          <div className="avatar">
            <img alt='Doug'
            src="http://www.croop.cl/UI/twitter/images/doug.jpg" />
            Doug
          </div>
          <span className="time">
            An hour ago
          </span>
          <p>Ate lunch</p>
          <div className="commentCount">
            2
          </div>
        </div>
        {/* ... */}
      </div>
    )
  }
}
```

```
class App extends React.Component {
  render() {
    return (
      <div className="notificationsFrame">
        <div className="panel">
          <Header />
          <Content />
        </div>
      </div>
    )
  }
}
```

Props and State

- React allows us to send data to a component with the same syntax of HTML attributes or properties on the component
- Let see example of sending data from parent component to child component by using props

```
class Header extends React.Component {
  render() {
    return (
      <div className="header">
        <div className="menuIcon">
          <div className="dashTop"></div>
          <div className="dashBottom"></div>
          <div className="circle"></div>
        </div>

        <span className="title">
          {this.props.title}
        </span>

        <input
          type="text"
          className="searchInput"
          placeholder="Search ..." />

        <div className="fa fa-search searchIcon"></div>
      </div>
    )
  }
}
```

```
{
  timestamp: new Date().getTime(),
  text: "Ate lunch",
  user: {
    id: 1,
    name: 'Nate',
    avatar: "http://www.croop.cl/UI/twitter/images/doug.jpg"
  },
  comments: [
    { from: 'Ari', text: 'Me too!' }
  ]
}
```

```
class Content extends React.Component {
  render() {
    const {activity} = this.props; // ES6 destructuring

    return (
      <div className="content">
        <div className="line"></div>

        {/* Timeline item */}
        <div className="item">
          <div className="avatar">
            <img
              alt={activity.text}
              src={activity.user.avatar} />
            {activity.user.name}
          </div>

          <span className="time">
            {activity.timestamp}
          </span>
          <p>{activity.text}</p>
          <div className="commentCount">
            {activity.comments.length}
          </div>
        </div>
      </div>
    )
  }
}
```

```
// these lines do the same thing
const activity = this.props.activity;
const {activity} = this.props;
```

Array of object

```
const activities = [
  {
    timestamp: new Date().getTime(),
    text: "Ate lunch",
    user: {
      id: 1, name: 'Nate',
      avatar: "http://www.croop.cl/UI/twitter/images/doug.jpg"
    },
    comments: [{ from: 'Ari', text: 'Me too!' }]
  },
  {
    timestamp: new Date().getTime(),
    text: "Woke up early for a beautiful run",
    user: {
      id: 2, name: 'Ari',
      avatar: "http://www.croop.cl/UI/twitter/images/doug.jpg"
    },
    comments: [{ from: 'Nate', text: 'I am so jealous' }]
  },
]
```

```
class Content extends React.Component {
  render() {
    const {activities} = this.props; // ES6 destructuring
    return (
      <div className="content">
        <div className="line"></div>
        {/* Timeline item */}
        {activities.map((activity) => {
          return (
            <div className="item">
              <div className="avatar">
                <img
                  alt={activity.text}
                  src={activity.user.avatar} />
                {activity.user.name}
              </div>
              <span className="time">
                {activity.timestamp}
              </span>
              <p>{activity.text}</p>
              <div className="commentCount">
                {activity.comments.length}
              </div>
            </div>
          );
        })}
      </div>
    )
  }
}
```

State

- However, React doesn’t allow us to modify the props that is sent from parent component for a good reason. You won’t know what is the value of props sent by parent if a child allow to modify it
- The child component may have it’s own state, to handle this, React give us ability to hold a state of the component
    - Similar to props, state can be accessed via this.state.[stateName]. Whenever the state changes (via this.setState function), the component will rerender