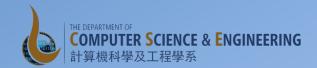
# React Components: State and Props

Jogesh K. Muppala





### State

- Each component can store its own local information in its "state"
  - Private and fully controlled by the component
  - Can be passed as props to children
- Only class components can have local state

#### State

 State declared within the constructor:

```
class Menu extends Component {
     constructor(props) {
          super(props);
          this.state = {
               selectedDish: null
```

State should only be modified using setState()

```
onDishSelect(dish) {
    this.setState({
        selectedDish: dish
    });
}
```

Never do the following:

```
this.state.selectedDish = dish;
```

### Props

- JSX attributes are passed into a component as a single object
  - Available in the component as "props"
  - Can pass in multiple attributes
  - Cannot modify props within the component
- Examples:
  - <Menu dishes={this.state.dishes} />
  - Here the dishes are available as props within the Menu Component and can be accessed as this.props.dishes
  - <Dishdetail dish={this.state.dish} comments={this.state.comments} />
  - Here dish is available as props within the Dishdetail Component and can be accessed as this.props.dish, and comments as this.props.comments

# **Handling Events**

- Handling events is similar to the way you handle events on DOM elements:
  - Use camelCase to specify events
  - Pass function as the event handler
- Example:

```
<Card onClick={() => this.onDishSelect(dish)}>
```

# Lifting State Up

- Sometimes several components may share the same data
- Changes to data in one component needs to be reflected to another component
- Best to move the shared state to a common ancestor component

## Lists and Keys

- Lists are handled similar to JavaScript
- Example:

- Keys should be given to elements inside the array
  - Help identify which items have changed, are added or removed