ReactJS
State and Props

Agenda

- Components
- State
- Props
- Props vs State
- Default Props
- Validating Props
- Nested Component
- Exercise

State

- Each component can store it’s own local information in it’s state

  - Private and fully controlled by the component

  - Can be passed as props to children

  - Its scope is limited to the current component.

- Only class component can have local state

```
const MenuItem = (props) => {
    //can not have state.
}
```

State

- State declared within the constructor:
- State should only be modified by setState:
- Never do the following: this.state.selectedDish = dish;
- The state of the parent component usually ends up being props of the child component

State

Example : https://github.com/priya1091992/React--State-and-Props

State

Default State -> ES5

```
var MyComponent = React.createClass({
getInitialState() {
return { /* initial state */ };
 },
});
```

State

Default State -> ES6

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.state = { /* initial state */ };
  }
}
```

State

State modification and component updates

```
class example extends React.Component{
  constructor(props){
    super(props);
      this.state={
        inputValue: " "
      }
  }
  onChange(e) {
     this.setState({ inputValue: e.target.value });
  }
```

State

State modification and component updates

```
render() {
    return (
        <input
            type='text'
            value={this.state.inputValue}
            onChange={this.onChange} />
    )}
}
```

Whenever this.state is updated the component will be re-rendered, causing the input value to reflect what the user typed.

Props

- Props (short for properties) are a Component's configuration.
- Received from parent and immutable.
- A Component cannot change its props, but it is responsible for putting together the props of its child Components.
- Props do not have to just be data -- callback functions may be passed in as props.

Props

props are received from a parent component and are **read only**

Props

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

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

Default Props

- To set the default props use getDefaultProps in the component. (ES5)

```
var Greeting = React.createClass({
        propTypes: {
        name: React.PropTypes.string
},
getDefaultProps: function() {
        return {
                name: 'Mary'
        };
},

});
```

Default Props

- To set the default props set the class.defaultProps in the component. (ES6)

```
class Greeting extends React.Component {
  // ...
}
Greeting.defaultProps = {
  name: 'Mary'
};
```

Validating Props

- We can also specify what type of props a component is expecting by specifying the **proptypes**.
- Syntax :
- <ClassName>.proptypes = {
- <propName> : <propType>
- }

Validating Props

If types does not match, react warns us about react validation failing.

```
App.propTypes = {
    propArray: React.PropTypes.array.isRequired,
    propBool: React.PropTypes.bool.isRequired,
    propFunc: React.PropTypes.func,
    propNumber: React.PropTypes.number,
    propString: React.PropTypes.string,
    propObject: React.PropTypes.object
}
```

```
▶ Warning: Failed propType: Invalid prop `propArray` of type `number` supplied to
  `App`, expected `array`.
```

Changing props and state

	props	state
Can get initial value from parent Component?	Yes	Yes
Can set default values inside Component?*	Yes	Yes
Can change inside Component?	No	Yes
Can set initial value for child Components?	Yes	Yes
Can change in child Components?	No	No

- Note that both props and state initial values received from parents override default values defined inside a Component.

Nested Component

```
import BankAccountsSearch from './BankAccountsSearch'
```

Nested Components are exactly what the name implies. They are children nested inside parent components. They simply help us create more complex UI structures. For example, an alert component probably has another sub-component nested within it to indicate the actual number of alerts.

Example: http://www.reactjstutorial.net/nested-components.html

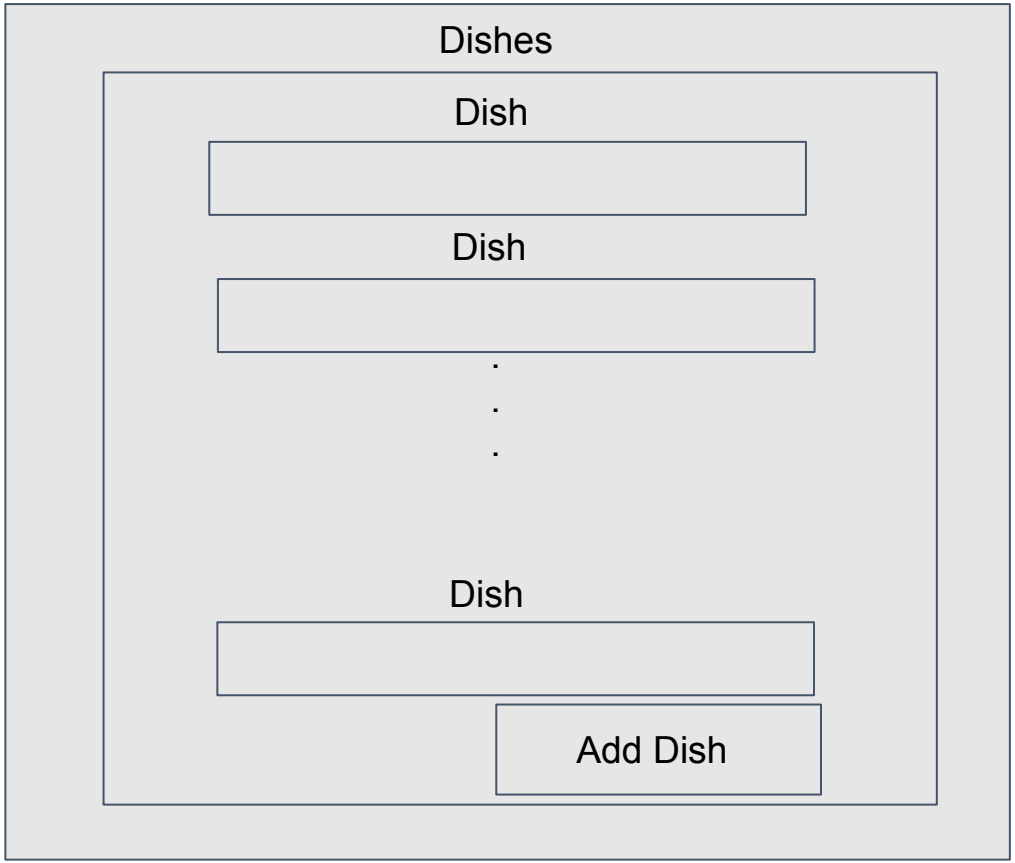
Exercise-State Manipulation

- implement a page showing 3 counters
- every counter has a + and a - button
- for every counter, clicking on + will increase the count number on that single counter
- for every counter, clicking on - will decrease the count number on that single counter
- the page will have 2 extra buttons "decrease all" and "increase all"
- clicking on "increase all" will increase the count by one unit on all the counters
- clicking on "decrease all" will decrease the count by one unit on all the counters

Exercise-Props

- Create MenuContainer class that will have dishes list in it’s own state. Pass that list in a child Dishes Component
- Render Dishes list(each dish should be rendered as a separate component like Dish) from Dishes component.
- On clicking of Dish card -> console additional details of particular dish like cost, etc..
- Add a button named Add Dish inside Dishes component after dish list
- On clicking of Add button, It should add a dish.

Exercise-Props



Your Queries Please!!!