# React

Lecture 2

# ES6

# Class

- What is a class?
- Constructor
- Methods
- Inheritance Example

# Class: Exercise

- Create a Vehicle Class with the following methods and properties:
    - type -> ex. “BUS”
    - wheels -> ex. 4
- Extend Vehicle for Car Class with a new property
    - name -> ex. “Wolkswagon”
- (Override the description method)

# (React) Components

# Components: Breaking down a page

- Break down a page into smaller parts.
- What should be a component?
  - Anything that will be used in multiple places/pages.


# Parts of a simple Task Manager App

# Components: Creating one

```
1  class Header extends React.Component{
2    render() {
3      return <h1>Best Task Manager</h1>;
4    }
5  }
6
7  const template = (
8      <div>
9          <Header/>
10     </div>
11 )
```

# Components: Exercise

- Create a component called `<Tasks />` that just has the following text in it:
  “This is the tasks component”

- Wrap all the components using `<App />`

# Components: Props

```
1  class Header extends React.Component{
2    render() {
3      // console.log(this.props)
4      return <h1>{this.props.title}</h1>;
5    }
6  }
7
8  class TodoApp extends React.Component{
9    render(){
10     const title = "Best Task Manager"
11     return(
12       <div>
13           <Header title={title}/>
14       </div>
15     )
16   }
}
```

# Components: Props (Exercise)

- Render each task as a `<Task />` for the passed `tasks` array.

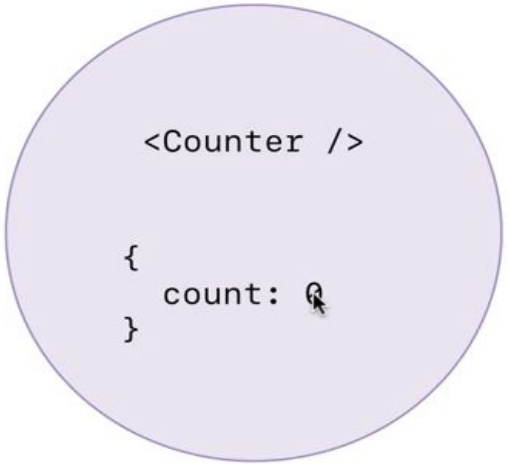
# Components: Events

- We define event handlers inside the class
- We use `this.handlerName` to call the handler
- We bind handlers in Constructor

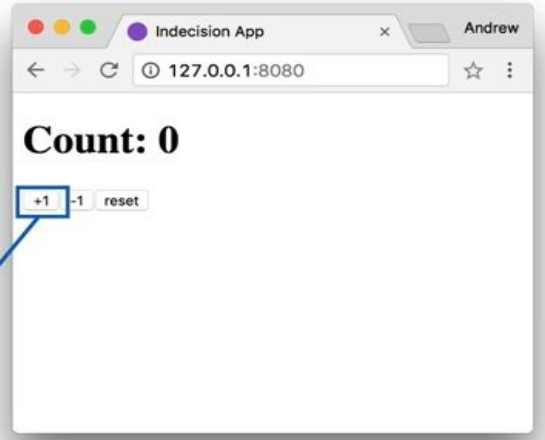
# Components: Events (Example)

```
1  class AddTask extends React.Component{
2
3    handleSubmit(e) {
4      e.preventDefault();
5      console.log("Submitted");
6    }
7
8    render(){
9      return(
10       <div>
11         <form onSubmit={this.handleSumbit}>
12             <input />
13             <button>Add Task</button>
14         </form>
15       </div>
16     )
     }
}
```

# React Component State



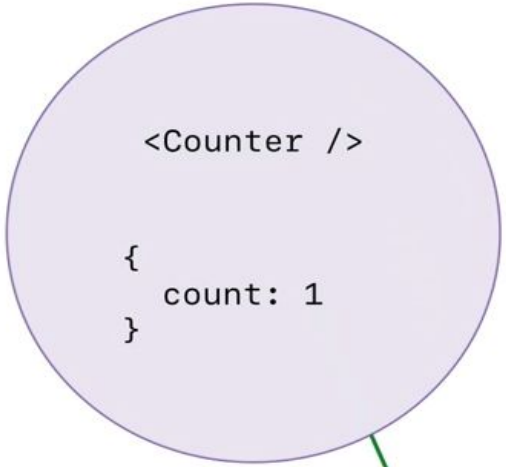

render



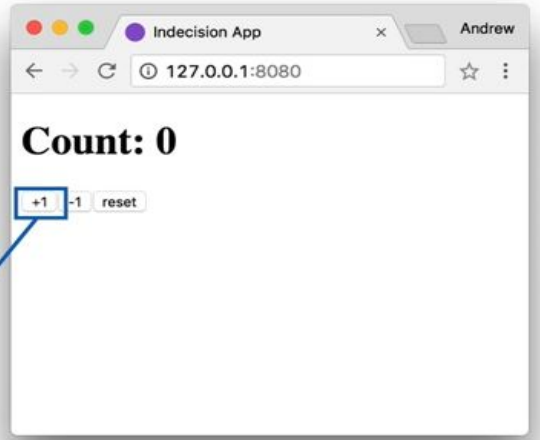

1. Setup default state object
2. Component rendered with default state values *

```
handleAddOne() {
// Increase "count" state by 1
}
```

# React Component State



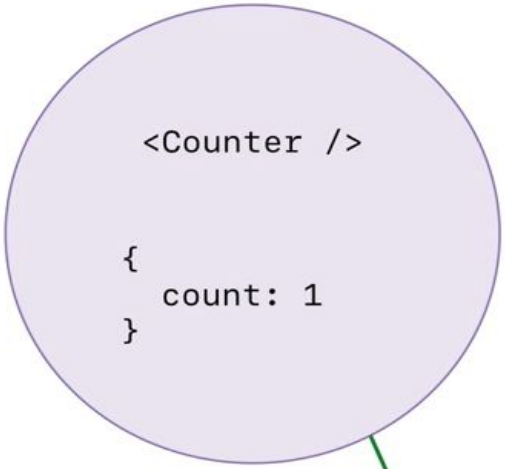

render



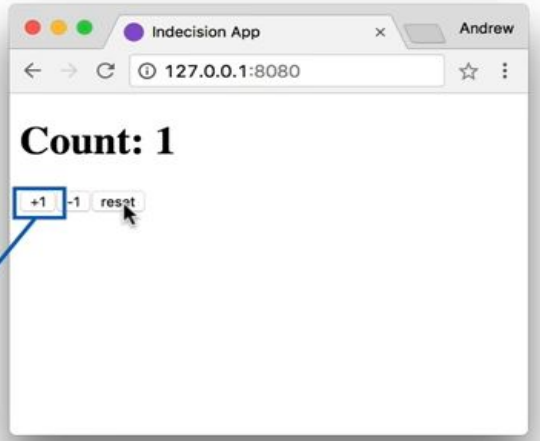

1. Setup default state object
2. Component rendered with default state values *
3. Change state based on event

```
handleAddOne() {
// Increase "count" state by 1
}
```

# React Component State




re-render




```
handleAddOne() {
// Increase "count" state by 1
}
```

1. Setup default state object
2. Component rendered with default state values *
3. Change state based on event
4. Component re-rendered using new state values *
5. Start again at 3

```jsx
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.handleAddOne = this.handleAddOne.bind(this);
    this.handleMinusOne = this.handleMinusOne.bind(this);
    this.handleReset = this.handleReset.bind(this);
    this.state = {
      count: 0
    };
  }
  handleAddOne() {
    this.setState((prevState) => {
      return {
        count: prevState.count + 1
      };
    });
  }
  handleMinusOne() {
    console.log('handleMinusOne');
  }
  handleReset() {
    console.log('handleReset');
  }
  render() {
    return (
      <div>
        <h1>Count: {this.state.count}</h1>
        <button onClick={this.handleAddOne}>+1</button>
        <button onClick={this.handleMinusOne}>-1</button>
        <button onClick={this.handleReset}>reset</button>
      </div>
    );
```

# Component: State

Only Updates the parts need to be updated, not the entire state.

```
this.state = {
  count: 0,
  name: 'Julie'
};
```

```
handleAddOne() {
  this.setState((prevState) => {
    return {
      count: prevState.count + 1
    };
  });
}
```

# Component: State (Exercise)

- Implement functionality for `handleMinusOne` function using state