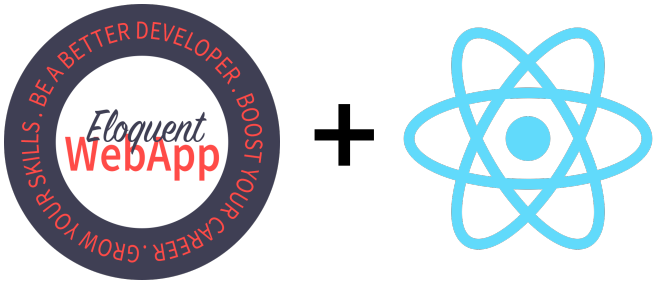
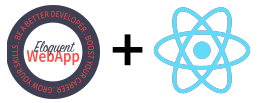



EloquentWebApp.com presents

# React.js Fundamentals



## RENDERING

- **ReactDOM.render(element, container)**

Render a ReactElement into the specified DOMElement, container. E.g.

```
// The ES5 way
ReactDOM.render(
        React.createElement(App, {name:
'EloquentWebApp'}),
        document.querySelector('.container')
);
```

```
// The ES5 + JSX way
ReactDOM.render(
        <App name='EloquentWebApp' />,
        document.querySelector('.container')
);
```

## CLASSES

- **React.createClass(specification)**

Creates a ReactClass given the specification object. E.g.

```
// The ES5 way
var App = React.createClass({
        /* optional lifecycle methods and
property initializers… */

        /* the render method is required */
        render: function () {
          return React.createElement('h1', null,
'Hello, React!');
        };
});
```

```
// The ES5 + JSX way
var App = React.createClass({
        render: function () {
          return <h1>Hello, React</h1>;
        };
});
```

- **class … extends React.Component {}**

Defines a ES6 class that extends React.Component. E.g.

```
// The ES6+ way
class App extends React.Component {
        render() {
          return <h1>Hello, React!</h1>;
        }
}
```

## PROPERTY INITIALIZERS

- **getDefaultProps**

Set values to this.props if props are not specified by the parent component.

- **getInitialState**

Set values as the initial value of this.state.

- **propTypes**

Validates props being passed to the component.

```
// The ES5 way
var App = React.createClass({
        getDefaultProps: function () {
          return { … };
        },
        getInitialState: function () {
          return { … }
        },
        propTypes: { … };
});
```

```
// The ES6+ way
class App extends React.Component {
        static defaultProps = { … }
        static  propTypes = { … }
        state = { … }
}
```

## PROPERTY VALIDATION

Validators available under React.PropTypes.

| | | |
|---|---|---|
| **array** | **bool** | **func** |
| **number** | **object** | **string** |
| **node** | **element** | |

```
instanceOf(Message)
oneOf(['News', 'Photos'])
oneOfType([validator, validator])
```

Optionally chain any of the above validators with `isRequired`.

## LIFECYCLE METHODS

- componentWillMount()

With ES6+, the class constructor now assumes the role previously filled by componentWillMount. E.g.

```
// The ES5 way
var App = React.createClass({
        componentWillMount: function () { … },
});
```

```
// The ES6+ way
class App extends React.Component {
        constructor(props) {
          super(props);
          // Manually bind methods to the component instance…
          this.handleButtonClick = this.handleButtonClick.bind(this);
        },
        handleButtonClick(e) {
          // ...to ensure that 'this' refers to the component instance here.
          this.setState({showText: true});
        }
}
```

- componentDidMount()
- componentWillReceiveProps(nextProps)
- shouldComponentUpdate(nextProps, nextState)-> bool
- componentWillUpdate(nextProps, nextState)
- componentDidUpdate(prevProps, prevState)
- componentWillUnmount()

## SPECIAL PROPS

- key

Uniquely identifies a React component.

- ref

Set a reference to the component's backing instance. It can be accessed through this.refs.

- className

The HTML class attribute.

- htmlFor

The HTML for attribute.

- style

Inline styles specified with an object. Style keys are camelCased.

- children

Parent component can read its children by accessing this.props.children. Use React.Children utilities to deal with it.

## COMPONENT API

- this.refs

Lists component instances with a ref prop.

- this.props

Contains any props passed to a component instance. Props are immutables.

- this.state

Contains the current state of a component instance set by getInitialState() or setState(). State are mutables.

- this.setState(nextState)

Performs an update of this.state with nextState, and re-renders the component.

- this.forceUpdate()

Re-renders the component.