React Tips & Tricks

React

PureComponent

```
class MyComponent extends React.PureComponent
```

```
shouldComponentUpdate(nextProps, nextState) {
    return (
        !shallowEqual(this.props, nextProps) ||
        !shallowEqual(this.state, nextState)
    );
}
```

PureComponent

Use PureComponent to avoid unnecessary render calls.

- list items
- connected components (Redux)
- independent sections of an application

Overhead is low.

PureComponent

```
const MyComponent = React.memo((props) => {
  /* */
});
```

"Complex" Render Functions

Avoid passing objects to children that fail the shallow compare of PureComponent.

- avoid inline event handlers (arrow functions)
- avoid creating objects
- avoid inline data manipulations
- simple conditionals and primitive results are acceptable

This will trigger a render function of MyComponent...

```
<MyComponent
    someOptions={{
        foo: true,
        bar: "test"
    }}
/>
```

Even with PureComponent.

Large Lists

- consider virtualization libraries for large sets of data (react-virtualized)
- react-dnd doesn't scale well with lots of targets (html5 dnd)

Large Components

Avoid monolithic components with large render functions.

Favor small, pure components.

But…

Measure. Measure. Measure.

React Developer Tools (Profiler) on Chrome.

Redux

Mutations on Reducers

- avoid deep clones
- shallow clone up to the property being changed

```
// state.list[index].options.name = "Test"

state.list = [ ...state.list ];

state.list[index] = {
    ...state.list[index],
    options: {
        ...state.list[index].options,
        name: "Test"
    }
};
```

Actions

Improve readability and functionality by using...

- small actions
- more redux-thunk
- action names as constants

Styled Components

Reusable Components

- use un-opinionated styling: avoid fixed sizes/margins/offsets
- provide a 'className' prop...

className

```
class MyComponent extends React.Component {
    render() {
        return {
            <div className={this.props.className}>
                <a href="#">test</a>
            </div>
        }
    }
}

const MyComponentStyled = styled(MyComponent)`
    background-color: red;
`;
```

Extending

```
const Button = styled.button`
    border-radius: 3px;
    color: gray;
`;
const RedButton = styled(Button)`
    color: red;
`;
```

Referring

```
const Link = styled.`
    color: blue;
`;
const Section = styled.div`
    ${Link}:hover {
        color: red
    }
`;
```

FIN