EVENT SWITCH.

LAYOUT COMPONENT.

CONTAINER COMPONENT.

by Anastasiia Derkach

softserve

EVENT SWITCH

What does it mean?

When writing event handlers it's common to adopt the handle{eventName} naming convention.

```
handleClick(e) { /* do something */ }
```

For components that handle several event types, these function names can be repetitive.

Example

```
handleEvent({type}) {
  switch(type) {
    case "click":
      return require("./actions/doStuff")(/* action dates */)
    case "mouseenter":
      return this.setState({ hovered: true })
    case "mouseleave":
      return this.setState({ hovered: false })
    default:
      return console.warn(`No case for event type "${type}"`)
  }
}
```

Alternatively, for simple components, you can call imported actions/functions directly from components, using arrow functions.

```
<div onClick={() => someImportedAction({ action: "DO_STUFF" })}
```

LAYOUT COMPONENT

What does it mean?

Layout components result in some form of static DOM element. It might not need to update frequently, if ever.

Consider a component that renders two children side-by-side. We can aggressively optimize this component.

```
<HorizontalSplit
  startSide={<SomeSmartComponent />}
  endSide={<AnotherSmartComponent />}
/>
```

While HorizontalSplit will be parent to both components, it will never be their owner. We can tell it to update never, without interrupting the lifecycle of the components inside.

```
class HorizontalSplit extends React.Component {
  shouldComponentUpdate() {
    return false;
  }

  render() {
    return (
      <FlexContainer>
        <div>{this.props.startSide}</div>
        <div>{this.props.endSide}</div>
      </FlexContainer>
    );
  }
}
```

CONTAINER COMPONENT

What does it mean?

A container does data fetching and then renders its corresponding sub-component. In the container's render method is where you compose your UI consisting of presentational children components. In order to have access to all stateful API's, a container must be a class component as opposed to a functional component.

```
const CommentList = ({ comments }) => (
  <ul>
    {comments.map(comment => (
      <li>
        {comment.body}-{comment.author}
      </li>
    ))}
  </ul>
);
```

Container Component

- Are concerned with how things work.
- May contain both presentational and container components** inside but usually don't have any DOM markup of their own except for some wrapping divs, and never have any styles.
- Provide the data and behavior to presentational or other container components.
- Call Flux actions and provide these as callbacks to the presentational components.
- Are often stateful, as they tend to serve as data sources.
- Are usually generated using higher order components such as connect() from React Redux, createContainer() from Relay, or Container.create() from Flux Utils, rather than written by hand.

We can create a new component responsible for fetching data and rendering the CommentList function component.

```
class CommentListContainer extends React.Component {
  constructor() {
    super()
    this.state = { comments: [] }
  }

  componentDidMount() {
    $.ajax({
      url: "/my-comments.json",
      dataType: 'json',
      success: comments =>
        this.setState({comments: comments});
    })
  }

  render() {
    return <CommentList comments={this.state.comments} />
  }
}
```

Why to use?

- We've separated our data-fetching and rendering concerns.
- We've made our CommentList component reusable.
- We've given CommentList the ability to set PropTypes.
- We can write different containers for different application contexts.

Container components are part of a strategy of separating responsibility between high-level and low-level concerns. Containers manage things like subscriptions and state, and pass props to components that handle things like rendering UI.
HOCs use containers as part of their implementation. You can think of HOCs as parameterized container component definitions.

RESOURCES USED

- https://reactpatterns.com/#layout-component
- https://www.freecodecamp.org/news/the-best-way-to-bind-event-handlers-in-react-282db2cf1530/
- https://medium.com/@learnreact/container-components-c0e67432e005
- https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0