Frontend Development with React

Kirk Sullivan

Week #2: Props, State, and Components

Topics we will cover:

- Creating and Using functional components
- How to pass and use props in functional components
- How to pass state and use it in functional components
- How to create and invoke anonymous functions

Components

- **Components are the fundamental building block for React**
- **Have internal state and external props**
- **Can be nested inside each other and used across multiple files and even projects**
- **Always start with a capital letter**

You can compare components with lego blocks. We use them as building blocks to build a bigger meaningful application

Two Main Types of Components

Class Component
AKA a Stateful Component

```
class App extends Component {
  constructor() {
    super();
    this.state = {
      name: 'React'
    };
  }

  render() {
    return (
      <div>
        <Hello name={this.state.name} />
        <p>
          Start editing to see some magic happen :)
        </p>
      </div>
    );
  }
}
```

These are your traditional React Components. Specifically when you need a component that needs a lifecycle method and/or State you want to use this type of component.

State

- **Data is called “state”**
  - **More precisely “state” is data that changes**
- **State is always an object with key value pairs**
- **Is always initialized in the constructor**

- **When you update “state”, React re-renders the view for you**
- **State is a keyword.**

```
constructor() {
  super();
  this.state = {
    name: 'React'
  };
}
```

Two Main Types of Components (continued)

Functional Component
AKA a Stateless Component

```
1   import React from 'react';
2
3   const Color = (props) => {
4     const color = props.color
5     const selectColor = props.selectColor
6     return(
7       <div className ={props.color} onClick={() => selectColor(color)} />
8     )
9   }
10
11  export default Color
```

This function is a valid React component because it accepts a single "props" (which stands for properties) object argument with data and returns a React element. We call such components "function components" because they are literally JavaScript functions.

—React Documentation

Functional Component (continued)

Functional Component
AKA a Stateless Component

```
11    export default Color
```

To use our functional component throughout our application we have to export the component.

Props

Think of props as “options” passed to a component to customize its functionality.

Props are conceptually and syntactically very similar to an HTML property.

All Props are passed into a component become key-value pairs on that component’s “props” object.

```
  render() {
    return (
      <div>
        <Hello name={this.state.name} />
        <p>
          Start editing to see some magic happen :)
        </p>
      </div>
    );
  }
}
```

Anonymous functions

They’re are called anonymous functions because aren’t given a name.

This is a common strategy for passing arguments to event handlers in React - instead of passing the function directly to the click handler, we wrap it in another (often anonymous) function - this gives us more "room" to pass our own arguments to the function

```
3  const Color = (props) => {
4    const color = props.color
5    const selectColor = props.selectColor
6    return(
7      <div className ={props.color} onClick={() => selectColor(color)} />
8    )
9  }
10
```

Resources

https://reactjs.org/docs/components-and-props.html

GitHub Link to the workshop and files

https://github.com/Voodoobrew/workshop-2