Understanding State and Events

Overview

- React State
- Events
- Forms

State

State

Generally in application development, the word *“State”* *is* used to refer to the data that an application holds at a given time.

In other words, the data - variables, arrays, objects, etc - describes the state of your application .

NB: State can be used at different levels of the application and may convey slightly different meaning depending on the context

State and React.Component

State consists of any data your application needs to know about, that can change over time.

The idea is to let your app to respond to state changes and present an updated UI when necessary.

React provide a simple Object and method for state management in class components

Stateful Components - state = {}

The `state` property must be set to a JavaScript `object`.

```
class StatefulComponent extends React.Component {
        constructor(){
                state = { name:"Codetrain" }
        }
        render(){
                return <h1>Welcome to {this.state.name}</h1>
        }

}
```

Accessing State Data - *this.state.key*

As state is a property of the React.Component class and it must be accessed through the “this” keyword when being accessed from within a method of the class.

NB: in the previous example we used “*this*” to access state from the render() method

```
return <h1>Welcome to {this.state.name}</h1>
```

Mutating State

React state values can be changed using the setState method

```
this.setState({name: “Codetrain Ghana”})
```

We can also pass a function that returns an object to setState

```
this.setState( (newName)=> {name: newName} )
```

We will implement these techniques soon

Events

Events

React elements come with all the appropriate event handlers we are used to in the realms of HTML.

React events are named using camelCase, rather than lowercase.

With JSX you pass a function as the event handler, rather than a string.

Events

```
<button onClick={IncreaseByOneEventListner} >

        Increase by One

</button>
```

Changing State with Events - An example

```
import React from “react”;

class CounterComponent extends React.Component{

        state = { counter : 0 }
        render(){

        return <button onClick={()=> this.setState({counter: this.state.counter
+ 1})}>Add One</button>

        }

}
```

Forms, Events, and State

Controlled elements

Form control elements such as input,radio,textArea, e.t.c can be linked to react component state such that when their values change the state is updated by taking advantage of events.

Event, Event listener and State

To link the a input element to component state, You need:

- Event - is the event you want to listen to on the input element eg: onChange
- Event listener - the method called when Event occurs
- State - component state key to updated

State and Event listener

```
state={ inputText: “” }

onChangeEventListener(event){

      this.setState({inputText: event.target.value})

}
```

The controlled input

```
<input value={this.state.inputText} onChange={(event) =>
this.onChangeEventListener(event)} />
```

Let’s do some practice!

Exercise 1: instructions

1. Update your projects from week 2 to use State

2. Replace parts of the returned JSX to access the state object using this.state

Exercise 2: instructions

1. Create a new react app
2. Create a stateful component
3. Render a form with an input box
4. Associate the input box to the local state to store the value of the input box on submit