React Basics II

Objectives

- [ ] Describe the process for writing a component
- [ ] Write a simple stateful component
- [ ] Understand how to set up event listeners

React Basics I Recap (JSX)

- JSX is the stuff that looks like HTML
- Any expression can go in JSX if put inside of {}
- JSX is nestable

```
<p>I'm JSX</p>
```

```
const myName = "Dylan";
<p>{myName}</p>
```

```
<article>
    This is JSX
    <p>I'm JSX too</p>
    <MyCustomMadeComponent/>
</article>
```

React Basics I Recap (Components)

- Components are functions that usually return JSX
- You don't invoke them, you use JSX syntax
- You can pass information from one component to another using props. Props can be primitives, reference types, other components, etc.
- Props are passed as an argument and accessed like any JS object

```
const MyComp = () => <p>I'm a component</p>
```

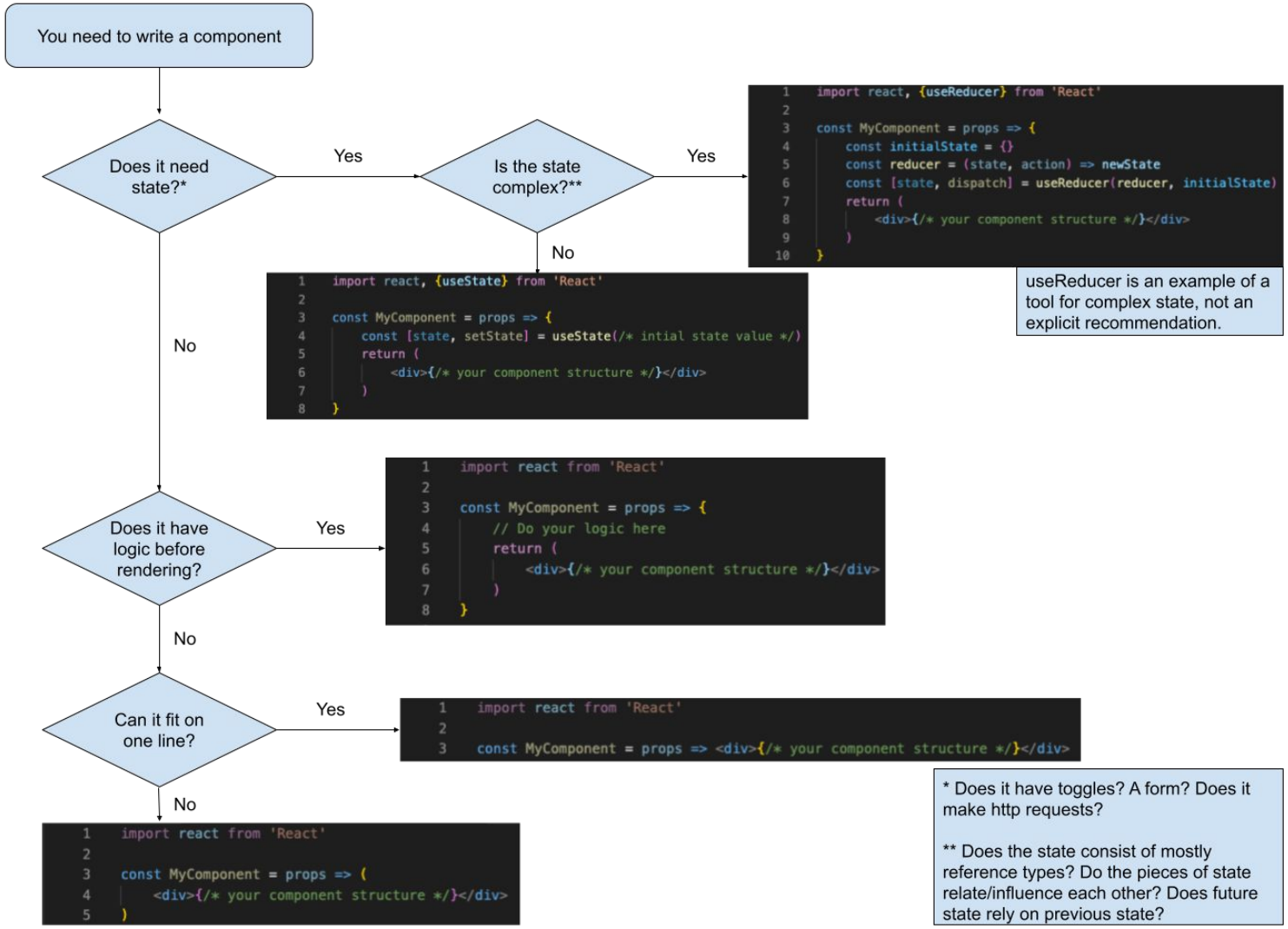
```
<MyComp/>
```

```
const luckyNumber = 7;
<MyComp luckyNumber={7} />
```

```
const MyComp = props => <p>My number is {props.luckyNumber}</p>
```

How do I write the right component?

- Don’t use a class component
- Use the right kind of state
- Be terse if you can
  - Don’t include parenths unless destructuring your props
  - Rely on implicit return when you can


You need to write a component
Does it need state?*
Yes
Is the state complex?**
Yes
No
No
Does it have logic before rendering?
Yes
No
Can it fit on one line?
Yes
No
import react, {useReducer} from 'React'
const MyComponent = props => {
const initialState = {}
const reducer = (state, action) => newState
const [state, dispatch] = useReducer(reducer, initialState)
return (
{/* your component structure */}
)
}
useReducer is an example of a tool for complex state, not an explicit recommendation.
import react, {useState} from 'React'
const MyComponent = props => {
const [state, setState] = useState(/* intial state value */)
return (
{/* your component structure */}
)
}
import react from 'React'
const MyComponent = props => {
// Do your logic here
return (
{/* your component structure */}
)
}
import react from 'React'
const MyComponent = props => {/* your component structure */}
import react from 'React'
const MyComponent = props => (
{/* your component structure */}
)
* Does it have toggles? A form? Does it make http requests?
** Does the state consist of mostly reference types? Do the pieces of state relate/influence each other? Does future state rely on previous state?

Stateless Components I

```
3   const MyComponent = props => {
4       // Do your logic here
5       return (
6           <div>{/* your component structure */}</div>
7       )
8   }
```

- Use curly braces
- conduct your logic
- use a return statement with any expressions wrapped in parentheses (if it’s a multiliner)

Stateless Components II

```
3   const MyComponent = props => (
4       <div>{/* your component structure */}</div>
5   )
```

- Use arrow functions’ implicit return

Stateless Components III

```
3   const MyComponent = props => <div>{/* your component structure */}</div>
```

- Don’t use parentheses if you can fit it on one line <80 chars

Stateful Components

```
3   const MyComponent = props => {
4       const [state, setState] = useState(/* intial state value */)
5       return (
6           <div>{/* your component structure */}</div>
7       )
8   }
```

- Import useState or use React.useState

Stateful Components

```
4    const [state, setState] = useState(/* intial state value */)
```

Stateful Components

- The brackets to the left of the assignment are **array destructuring**.

```
const myArr = ['apple', 'orange', 'banana', 'kiwi']
const [fruit1, fruit2, ...otherfruits] = myArr
console.log(fruit1) // "apple"
console.log(fruit2) // "orange"
console.log(otherFruits) // ['banana', 'kiwi']
```

```
3   const [state1, setState1] = useState()
4
5   // This is the exact same as above
6   const useStateArray = useState()
7   const state2 = useStateArray[0]
8   const setState2 = useStateArray[1]
```

Stateful Components

```
1   const [username, setName] = useState('Bill')
2
3   console.log(typeof username) // "string"
4   console.log(typeof setName) // "function"
5
6   const [age, setAge] = useState()
7
8   console.log(typeof age) // "undefined"
9   console.log(typeof setAge) // "function"
```

https://codepen.io/DMKite/pen/mdOaJoq?editors=0011

Adding Event Listeners

- onClick, onChange, onSubmit are props
- Each should be a function
- The function will take an event object

```
<button onClick={myFunction}>Do Thing</button>
```

```
<button onClick={e => console.log(e)}>Do Thing</button>
```

https://codepen.io/DMKite/pen/ZEBVRMZ?editors=1011

Non-primitive State

- Setting state does not carry any data over. You must spread state if you are changing a single key or array entry

```
const [state, setState] = useState({label: null, value: null})
console.log(state) // {label: null, value: null}
setState({label: 'Dog'})
console.log(state) // {label: 'Dog'}

const [state, setState] = useState({label: null, value: null})
console.log(state) // {label: null, value: null}
setState({...state, label: 'Dog'})
console.log(state) // {label: 'Dog', value: null}
```

Objectives

- [ ] Describe the process for writing a component
- [ ] Write a simple stateful component
- [ ] Understand how to set up event listeners