

# REACT HOOKS AND YOU

---

# AGENDA

- **useState**
  - Replaces this.setState
- **Rules**
  - Order Matters
  - Incompatible with Classes
- **useEffect**
  - Replaces componentDidMount, componentDidUpdate, componentWillUnmount



# SETTING STATE

## Class Component

```
class YourName extends React.Component {
  constructor() {
    super()
    this.state = { name: "" }
    this.handleChange = this.handleChange.bind(this)
  }
  handleChange(evt) {
    this.setState({ name: evt.target.value })
  }
  render() {
    return (
      <div>
        <h1>Name: {this.state.name}</h1>
        <input
          type="text"
          value={this.state.name}
          onChange={this.handleChange}
        />
      </div>
    )
  }
}
```

## Hooks Component

```
function YourName() {
  const [name, setName] = useState("")
  const handleChange = evt => {
    setName(evt.target.value)
  }
  return (
    <div>
      <h1>Name: {name}</h1>
      <input
        type="text"
        value={name}
        onChange={handleChange} />
    </div>
  )
}
```



# SETTING STATE

## Class Component

```
class YourName extends React.Component {
  constructor() {
    super()
    this.state = { name: "" }
    this.handleChange = this.handleChange.bind(this)
  }
  handleChange(evt) {
    this.setState({ name: evt.target.value })
  }
  render() {
    return (
      <div>
        <h1>Name: {this.state.name}</h1>
        <input
          type="text"
          value={this.state.name}
          onChange={this.handleChange}
        />
      </div>
    )
  }
}
```

## Hooks Component

```
function YourName() {
  const [name, setName] = useState("")
  const handleChange = evt => {
    setName(evt.target.value)
  }
  return (
    <div>
      <h1>Name: {name}</h1>
      <input
        type="text"
        value={name}
        onChange={handleChange} />
    </div>
  )
}
```

1. Initialize State
2. Use State
3. Set State



# SETTING STATE



- `import React, { useState }`  
from “react”
- `useState` takes initial state as argument
- `useState` returns an array
  - First element is the current value
  - Second element is a setter function
- No need to bind `handleChange`!
  - State is available as a variable in the same scope

## Hooks Component

```
import React, { useState } from "react"

function YourName() {
  const [name, setName] = useState("")
  const handleChange = evt => {
    setName(evt.target.value)
  }
  return (
    <div>
      <h1>Name: {name}</h1>
      <input
        type="text"
        value={name}
        onChange={handleChange} />
    </div>
  )
}
```



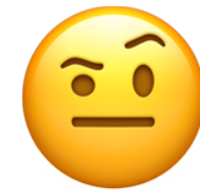


# RULES





# RULES: SETTING STATE



1. `useState` *cannot* be used in a class component!
2. `useState` must run *on every render*
  - Don't put it in an if-block!
  - Don't put it after a conditional return!
3. Get used to array destructuring!

## Hooks Component

```
import React, { useState } from "react"

function YourName() {
  const [name, setName] = useState("")
  const handleChange = evt => {
    setName(evt.target.value)
  }
  return (
    <div>
      <h1>Name: {name}</h1>
      <input
        type="text"
        value={name}
        onChange={handleChange} />
    </div>
  )
}
```



# RULES: SETTING STATE



1. `useState` *cannot* be used in a class component!
2. `useState` must run *on every render*
  - Don't put it in an if-block!
  - Don't put it after a conditional return!
3. Get used to array destructuring!

Clahooks(?) Component

```
class YourName extends React.Component {  
  render() {  
    const [name, setName] = useState("")  
    return (  
      <div>  
        <h1>Name: {name}</h1>  
        <input  
          type="text"  
          value={name}  
          onChange={evt =>  
            setName(evt.target.value)  
          }  
        />  
      </div>  
    )  
  }  
}
```





# RULES: SETTING STATE



1. `useState` *cannot* be used in a class component!
2. `useState` must run *on every render*
  - Don't put it in an if-block!
  - Don't put it after a conditional return!
3. Get used to array destructuring

## Hooks Component

```
function YourName(props) {  
  if (props.dontRender) {  
    return <div>Not This Time!</div>  
  }  
  let name = ""  
  let setName = () => {}  
  if (props.someThing) {  
    [name, setName] = useState("")  
  }  
  const handleChange = evt => {  
    setName(evt.target.value)  
  }  
  return (  
    <div>  
      <h1>Name: {name}</h1>  
      <input  
        type="text"  
        value={name}  
        onChange={handleChange} />  
    </div>  
  )  
}
```



# RULES: SETTING STATE



1. `useState` *cannot* be used in a class component!
2. `useState` must run *on every render*
  - Don't put it in an if-block!
  - Don't put it after a conditional return!
3. Get used to array destructuring!

## Hooks Component

```
function YourName() {  
  const nameState = useState("")  
  const name = nameState[0]  
  const setName = nameState[1]  
  const handleChange = evt => {  
    setName(evt.target.value)  
  }  
  return (  
    <div>  
      <h1>Name: {name}</h1>  
      <input  
        type="text"  
        value={name}  
        onChange={handleChange} />  
    </div>  
  )  
}
```





# CODE DEMO TIME

