# Managing State with Refs

**Cory House**
REACT CONSULTANT AND TRAINER

@housecor reactjsconsulting.com

# Agenda

**What's a ref?**

**When to consider a ref**

**Controlled vs Uncontrolled components**

**Implement refs**

- Uncontrolled form inputs
- Track if the component is mounted
- Store a previous value

# What's a Ref?

Reference an HTML element

Store a value that's stable between renders

Can mutate the ref's value directly

Don't cause a re-render when they change

# Ref Example

```
import React, { useRef } from 'React';

function TextInputWithFocusButton() {
  const inputEl = useRef(null);
  const onButtonClick = () => inputEl.current.focus();

  return (
    <>
      <input ref={inputEl} type="text" />
      <button onClick={onButtonClick}>Focus the input</button>
    </>
  );
}
```

# When to Use a Ref

**DOM element reference**

**State that isn't rendered/doesn't change**

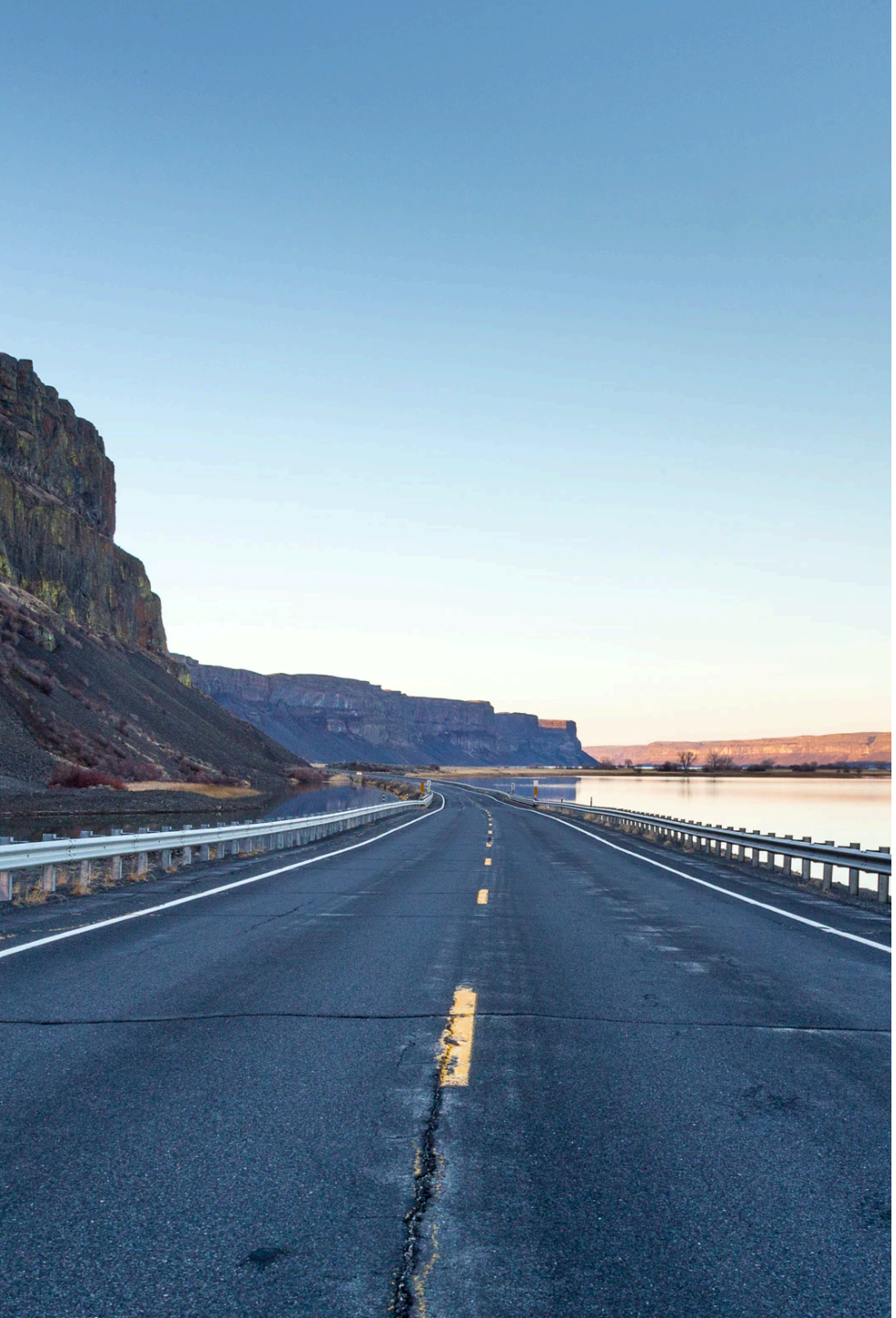
**"Instance variables" in func components:**

- Keep data between renders
- Storing a previous value
- Track if component is mounted
- Hold HTTP request cancel token
- Reference a 3rd party library instance
- Debounce a call / declare local cache
- Store flag that something happened
- Store value used in useEffect

# Uncontrolled vs Controlled

| | Uncontrolled | Controlled |
|---|---|---|
| Set initial value | ✓ | ✓ |
| Validate on submit | ✓ | ✓ |
| Validate instantly | | ✓ |
| Conditionally disable submit | | ✓ |
| Enforce input format | | ✓ |
| Several inputs for one piece of data | | ✓ |
| Dynamic inputs | | ✓ |

Source: goshakkk.name/controlled-vs-uncontrolled-inputs-react/



# When to Go Uncontrolled

**Extreme performance requirements**

**Working with non-React libraries**

# Summary

## When to use refs

- Reference a DOM element
- Manage uncontrolled inputs
- Store state that isn't rendered
- Extreme performance demands
- Instance variables in functions

## Next up: useReducer