

React – Forms

Controlled / Uncontrolled Form Components



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#react

Table of Contents

1. Uncontrolled Forms
2. Controlled Forms
3. useRef Hook
4. Validation and Error Handling





Forms

Controlled and Uncontrolled Forms

Forms

- Form elements work a little bit **differently** in React
 - They naturally keep some **initial state**
- React provides **2** standard ways to handle forms
 - **Controlled** components
 - **Uncontrolled** components



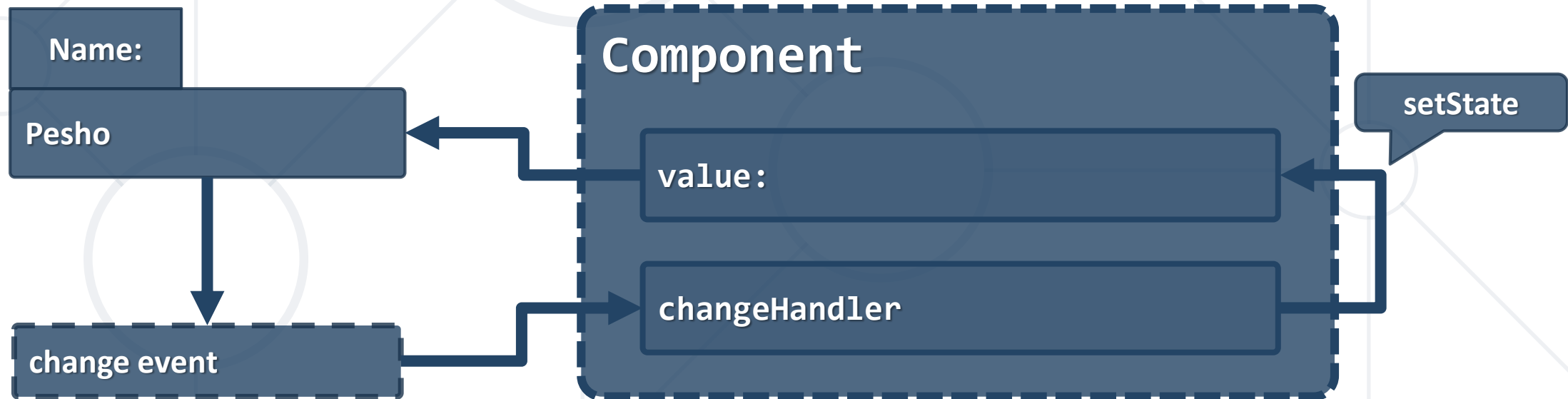
- Uncontrolled components provide a different approach to managing user input
- Values managed by the DOM
- Values accessed through refs, not React state
- When to Use **Uncontrolled** Components:
 - For simpler forms with fewer states.
 - In cases where direct DOM manipulation is needed.



Controlled Components

- Most **recommending** technique to implement forms
- In a **controlled component**, form data is handled by a **React**
 - Input element's value is kept in the **state**
 - Custom handlers for **change** or **submit** events
 - Piping all the input state through a React Component

- Inputs are **explicitly managed**
 - The **value** is stored inside the component **state**
 - **Input** is recorded with **event handlers**



Form Component Declaration

```
const Register = () => {  
  const [email, setEmail] = useState('');  
  
  const emailChangeHandler = (e) => {  
    setEmail(e.target.value);  
  }  
  
  submitHandler(event) {  
    event.preventDefault();  
    // Doing some AJAX with the data...  
  }  
  
  // Continues ...  
}
```

Form Component Rendering

```
...
return (
  <form onSubmit={submitHandler}>
    <div>
      <label htmlFor='email'>Email:</label>
      <input type="email"
        value={email}
        onChange={emailChangeHandler} />
    </div>
    // And so on for other input elements...
    <button type="submit">Register</button>
  </>
);
};
```

- React handles all form **input elements** the same
 - **textarea** uses a **value** prop

```
<form onSubmit={this.handleSubmit}>  
  <textarea value={this.state.value} onChange={this.handleChange} />  
  <input type="submit" value="Submit" />  
</form>
```

- **select** uses a **value** prop on the root element
 - Pass an array to select multiple values

```
<select multiple={ true } value={ ['A', 'B', 'C'] }>
```

- Sometimes can be tedious to use controlled components
 - Writing every handler for every way your data
 - Pipe all the input state
- Reuse **changeHandler** and **state**
- **Uncontrolled components** is an alternative technique for implementing input forms



useRef

- To write an Uncontrolled Component you can create a **reference** to specific DOM element
 - Refs are created using **React.createRef()**
 - Attached to React elements via the **ref** attribute

```
const MyComponent = () => {  
  myRef = React.createRef();  
  
  return <div ref={this.myRef} />;  
}
```

Using Refs Example

```
const Register = () => {  
  inputRef = React.createRef();  
  
  handleSubmit(event) {  
    event.preventDefault();  
    alert('A name was submitted: ' + inputRef.current.value);  
  }  
  // Continues ...  
}
```


Using Refs Example

```
render() {  
  return (  
    <form onSubmit={handleSubmit}>  
      <label>  
        Name:  
        <input type="text" ref={inputRef} />  
      </label>  
      <input type="submit" value="Submit" />  
    </form>  
  );  
}
```

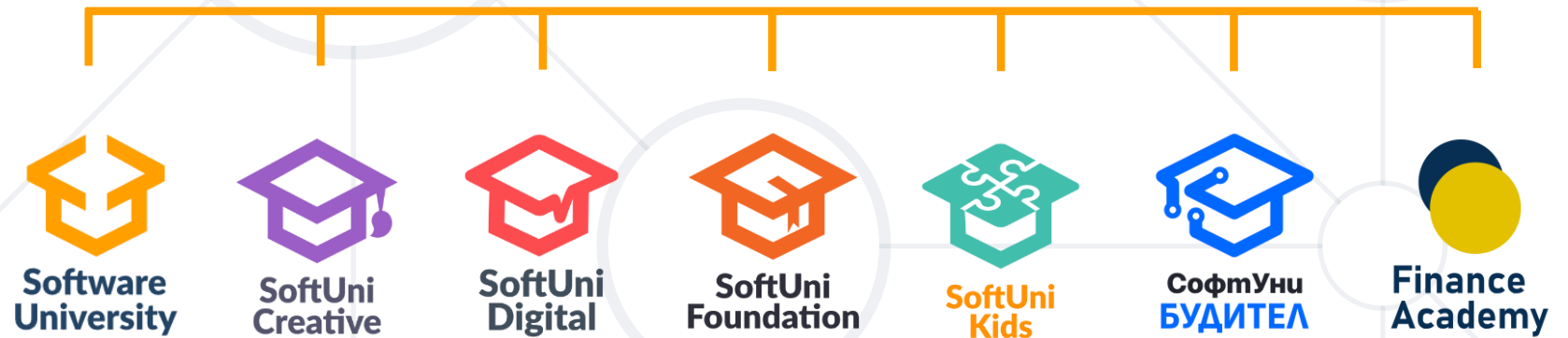
When to Use Refs?

- There are a few good **use cases** for refs
 - Managing focus, text selection, or media playback
 - Triggering **imperative animations**
 - Integrating with third-party DOM libraries
- Avoid using refs for anything that can be done **declaratively**
- Don't overuse refs

- Forms
 - Controlled Components - **Unified Input Approach**
 - Uncontrolled Components - **Using Refs**
 - Validation and Error handling



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

