Forms with React

Ustun Ozgur

2015-11-03 Tue

Forms

- Uncontrolled
- Controlled

Uncontrolled

- Discouraged
- Native DOM elements are the source of truth, not sate
- Values read from events or DOM elements through refs
- ref: unique identifier for a child
- `this.refs` an object like `props`, `state` that is an object of refs

Refs

- Example

```
<div><span ref="inner">This is the inner text</span></div>
```

- The inner span can be referred as `this.refs.inner`
- To retrieve actual DOM node, use:
  - `React.findDOMNode`: preferred
  - `this.refs.inner.getDOMNode`: deprecated
- To retrieve the value, use value after getting the DOM node
- `this.refs.inner.getDOMNode().value`
- As of React 0.14, no need for getDOMNode() for native elements
- `this.refs.inner.value`

Example: Form with Uncontrolled Inputs

```
<form onSubmit={this.onSubmit}>

<input ref={name} placeholder="What is your name"/>
<input ref={surname} placeholder="What is your surname?"/>
<button type="submit">Submit</button>

</form>
```

Example (cont'd)

```
var MyForm = React.createClass({
render: function () {

  return <form onSubmit={this.onSubmit}>
    <input ref="name" placeholder="What is your name"/>
    <input ref="surname"
     placeholder="What is your surname?"/>
    <textarea ref="comment" placeholder="Comment"/>
    <label><input ref="fromSweden"
            type="checkbox"/>Are you from Sweden?</label>
    <label>Gender:
    <select ref="gender">
    <option>Male</option>
    <option>Female</option>
    </select></label>
    <button type="submit">Submit</button>
```

Example (cont'd)

```
onSubmit: function (e) {
  e.preventDefault();
  var name = this.refs.name.value;
  var surname = this.refs.surname.value;
  var comment = this.refs.comment.value;
  var fromSweden = this.refs.fromSweden.checked;
  var gender = this.refs.gender.value;

  console.log("Form submitted with",
              name, surname, comment,
              fromSweden, gender);

}});

ReactDOM.render(<MyForm/>, document.body);
```

Observations

- For checkboxes, we use checked attribute, not value.
- textarea does not have children content unlike HTML. Its value can be retrieved/set from its value attribute.
- Forms have an onSubmit event handler, however this by default calls the default form handler as well. We need to preventDefault.
- Events are synthetic events that wrap and normalize native DOM events.

Controlled Forms

- Idiomatic way
- Each input component accepts a value or checked property
- Each input component exposes onChange
- Set the value from the state
- Update the state in onChange

Example: Controlled Input (initial state)

```
var MyForm = React.createClass({

  getInitialState: function () {
    return {name: ''};
  },
  render: function () {
    return <form onSubmit={this.onSubmit}>
      <input value={this.state.name} placeholder="What is yo
      </form>
  }
})
```

Example (cont'd) (the event handlers)

```
var MyForm = React.createClass({

  getInitialState: function () {
    return {name: ''};
  },
  changeName: function (e) {
    this.setState({name: e.target.value});
  },
  render: function () {
    return <form onSubmit={this.onSubmit}>
      <input onChange={this.changeName}
       value={this.state.name}
       placeholder="What is your name?"/>
      </form>}})
```

Example (the submit handler)

```
var MyForm = React.createClass({
  onSubmit: function () {
    console.log("the form values are", this.state.name); },
  getInitialState: function () {
    return {name: ''};  },
  changeName: function (e) {
    this.setState({name: e.target.value});  },
  render: function () {
    return <form onSubmit={this.onSubmit}>
      <input onChange={this.changeName}
       value={this.state.name}
       placeholder="What is your name?"/>
      </form>  }})
```

Other Events Related to Forms

- onBlur
- onFocus

A Debugging Trick

- A nice trick in debugging form is to output state visually
- `JSON.stringify(this.state, null, 4)` yields a properly indented version of state.

Example:

```
var MyForm = React.createClass({
  onSubmit: function () {
    console.log("the form values are", this.state.name); },
  getInitialState: function () {
    return {name: ''};  },
  changeName: function (e) {
    this.setState({name: e.target.value});  },
  render: function () {
    return <form onSubmit={this.onSubmit}>
      <input onChange={this.changeName}
       value={this.state.name}
       placeholder="What is your name?"/>
      <pre>{JSON.stringify(this.state, null, 4)}</pre>
      </form>  }})
```

DEMO

Exercise 1/2:

- Add another form input for credit card where the user can only enter numbers.

Hint: You can use `/^\d+$/.test(foo)` to test whether the variable ‘foo‘ consists of only numbers.

- Can you delete the card number after entering a few digits? If not, fix the bug. Hint: either change the regex or find some other means.
- As the user types in the name field, greet them with a gender prefix. The male names are: `["John", "George"]`. The female names are `["Jane", "Mary"]`.
- If gender cannot be determined from the name, greet with just the name. If there is no name yet, do not greet.

Exercise 2/2:

- As the user passes from the name field to the card field, validate the name such that it is at least 3 letters. If the name is 2 letters, show a warning. Note that you should not show the warning initially.

Hint: Think about state variables to keep track of. Should the variable that determines whether the input is valid or not be stored in state? Think about pros and cons.

- Add validation on submit. The name should be at least 3 letters, the card should be at least 3 digits. Should we be storing the validation of the form in state? Think about it.
- Implement adding a todo in the todo application.