React Forms

HTML form elemanları, React'te diğer DOM elemanlarından biraz farklı çalışır; çünkü form elemanlarının kendilerine has iç state'leri vardır. Örneğin, düz HTML'de yazılmış bu form, tek bir isim (name) değeri kabul etmektedir.

```
<form>
  <label>
    Name:
    <input type="text" name="name" />
  </label>
  <input type="submit" value="Submit" />
</form>
```

Bu form, kullanıcı formu gönderdiğinde yeni bir sayfaya göz atmak gibi varsayılan bir HTML formu davranışına sahiptir. Bu davranışı React'te istiyorsanız, zaten çalışmaktadır. Ancak çoğu durumda, formun gönderimini işleyen ve kullanıcının forma girdiği verilere erişen bir JavaScript fonksiyonuna sahip olmak uygundur. Bunu başarmanın standart yolu da "kontrollü bileşenler" adı verilen bir tekniktir.

* Kontrollü Bileşenler

HTML'de, <input>, <textarea> ve <select> gibi form elemanları genellikle kendi state'ini korur ve kullanıcı girdisine dayalı olarak güncellenir. React'te ise state'ler genellikle bileşenlerin this.state özelliğinde saklanır ve yalnızca setState() ile güncellenir.

React state'inde tek bir kaynak olarak ikisini birleştirebiliriz. Ardından form oluşturan React bileşeni, sonraki kullanıcı girişi üzerinde bu formda olanı da kontrol eder. Değeri React tarafından bu şekilde kontrol edilen bir girdi (input) form elemanına kontrollü bileşen denir.

Örneğin bir önceki örnekte, name girdisine yazılıp butona basıldığı zaman name'in değerini alert ile yazdırmak istiyorsak, formu kontrollü bir bileşen olarak oluşturabiliriz:

```
class NameForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {value: ''};
    this.handleChange = this.handleChange.bind(this);
```

```
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({value: event.target.value});
  }
  handleSubmit(event) {
    alert('A name was submitted: ' + this.state.value);
    event.preventDefault();
  }
  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input type="text" value={this.state.value} onChange={this.handleChange} />
        </label>
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

value özelleği input'un kendisinde zaten var. Öyleyse bu değeri almak için yeni bir React state'i oluşturmaya gerek yok. Bu inputta value olarak state'i yazdıracağız ve input'ta her değişiklik olduğunda bu state'i güncelleyeceğiz.

Kontrollü bir bileşende, input'un değeri her zaman React state'i tarafından kontrol edilir. Bu biraz daha fazla kod yazmanız anlamına gelse de; artık input'un değerini diğer UI elemanlarına gönderebilir, ya da başka olay yöneticileri ile sıfırlayabilirsiniz.

* Textarea Elemanı

HTML'de, <textarea> elemanı yazıyı alt elemanında tanımlar:

```
<textarea>
  Merhaba, burası textarea yazı alanıdır.
</textarea>
```

Bunun yerine React, <textarea> için bir value özelliği kullanır. Bu şekilde <textarea> kullanan bir form, tek satırlı bir girdi kullanan bir forma çok benzer şekilde yazılabilir:

```
class EssayForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: 'Bu kısma bir şeyler yazınız.'
    };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({value: event.target.value});
  }
  handleSubmit(event) {
    alert('Gönderilen değer: ' + this.state.value);
    event.preventDefault();
  }
  render() {
    return (
      <form onSubmit={this.handleSubmit}>
```

```
      <label>
        Essay:
        <textarea value={this.state.value} onChange={this.handleChange} />
      </label>
      <input type="submit" value="Gönder" />
    </form>
  );
 }
}
```

this.state.value'nun constructor'da başlatıldığına dikkat edin, böylece textarea içerisinde varsayılan olarak bu yazı bulunacaktır.

Select Elemanı

HTML'de <select>, bir açılır liste oluşturur. Örneğin, aşağıdaki kod bazı meyveleri listeler:

```
<select>
 <option value="grapefruit">Grapefruit</option>
 <option value="lime">Lime</option>
 <option selected value="coconut">Coconut</option>
 <option value="mango">Mango</option>
</select>
```

Coconut seçeneğinin başlangıçta selected özelliği yüzünden seçili olarak geleceğini unutmayın. React, bu selected özelliğini kullanmak yerine, select etiketinde bir value özelliği kullanır. Kontrollü bir bileşende bu daha kullanışlıdır çünkü yalnızca bir yerde güncelleme yapmanızı sağlar. Örneğin:

```
class FlavorForm extends React.Component {
 constructor(props) {
  super(props);
  this.state = {value: 'havuç'};
  this.handleChange = this.handleChange.bind(this);
```

```
  this.handleSubmit = this.handleSubmit.bind(this);
}
handleChange(event) {
  this.setState({value: event.target.value});
}
handleSubmit(event) {
  alert('Favori meyveniz: ' + this.state.value);
  event.preventDefault();
}
render() {
  return (
    <form onSubmit={this.handleSubmit}>
      <label>
        Favori meyveni seç:
        <select value={this.state.value} onChange={this.handleChange}>
          <option value="elma">Elma</option>
          <option value="armut">Armut</option>
          <option value="havuç">Havuç</option>
          <option value="muz">Muz</option>
        </select>
      </label>
      <input type="submit" value="Gönder" />
    </form>
  );
}
```

```
}
```

Genel olarak bu, <input type="text">, <textarea> ve <select> elemanlarının çok benzer şekilde çalışmasını sağlar.

> **Not**
> Bir `select` etiketinde birden fazla seçeneği seçmenize izin veren bir diziyi `value` özelliğine yazabilirsiniz:
>
> ```
> <select multiple={true} value={['B', 'C']}>
> ```

Okunamayan kod : <select multiple={true} value={['B', 'C']}>

* Dosya Girişi Elemanı

HTML'de bir <input type="file"> elemanı, kullanıcıyı cihazının depolama alanından bir veya daha fazla dosyayı sunucuya yüklemesini ya da JavaScript'in File API aracılığıyla manipüle etmesini sağlar.

```
<input type="file" />
```

Değeri salt okunur olduğu için, React'te kontrolsüz bir bileşendir. Daha sonra diğer kontrol edilemeyen bileşenlerle birlikte dokümanlarda ele alınmıştır.

* Birden Fazla Girdiyi Yönetmek

Birden fazla kontrollü input elemanını yönetmeniz gerektiğinde, her öğeye bir name özelliği ekleyebilir ve yönetici (handler) fonksiyonun event.target.name değerine dayanarak ne yapılacağını seçmesine izin verebilirsiniz.

Örneğin:

```
class Reservation extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      isGoing: true,
      numberOfGuests: 2
    };
```

```
  this.handleInputChange = this.handleInputChange.bind(this);
}
handleInputChange(event) {
  const target = event.target;
  const value = target.type === 'checkbox' ? target.checked : target.value;
  const name = target.name;
  this.setState({
    [name]: value
  });
}
render() {
  return (
    <form>
      <label>
        Is going:
        <input
          name="isGoing"
          type="checkbox"
          checked={this.state.isGoing}
          onChange={this.handleInputChange} />
      </label>
      <br />
      <label>
        Number of guests:
        <input
          name="numberOfGuests"
          type="number"
          value={this.state.numberOfGuests}
```

```
          onChange={this.handleInputChange} />
      </label>
    </form>
  );
 }
}
```

Verilen girdi ismine karşılık gelen state anahtarını güncellemek için ES6 syntax'ını nasıl kullandığımıza dikkat edin:

```
this.setState({
 [name]: value
});
```

Bu da ES5'teki eşdeğer kodudur.

```
var partialState = {};
partialState[name] = value;
this.setState(partialState);
```

Ayrıca, setState() otomatik olarak kısmi bir durumu geçerli duruma birleştirir olduğundan, yalnızca değiştirilen parçalarla çağırmamız gerekiyor.

* Kontrollü Girdilerde Null Değeri

Kontrollü bir bileşen üzerindeki value prop'unu belirtmek, sizin isteğiniz dışında kullanıcının girdi değerini değiştirmesini önler. value belirttiyseniz ancak girdi hala düzenlenebilir ise, yanlışlıkla value özelliğini undefined veya null olarak ayarlamış olabilirsiniz.

Aşağıdaki kod bunu göstermektedir. (Giriş ilk önce kilitlenir ancak kısa bir gecikme sonrasında düzenlenebilir hale gelir.)

```
ReactDOM.createRoot(mountNode).render(<input value="selam" />);
setTimeout(function() {
 ReactDOM.createRoot(mountNode).render(<input value={null} />);
}, 1000);
```

* Kontrollü Bileşenlere Alternatifler

Kontrollü bileşenleri kullanmak bazen sıkıcı olabilir, çünkü verilerinizin bir React bileşeniyle tüm giriş durumunu değiştirebilmesi ve yayınlayabilmesi için bir olay yöneticisi yazmanız gerekir. Bu özellikle,

önceden var olan bir kod tabanını React'e dönüştürürken veya bir React uygulamasını React olmayan bir kütüphaneyle birleştirirken can sıkıcı olabilir. Bu durumlarda, giriş formlarını uygulamak için alternatif bir teknik olan kontrolsüz bileşenler'i kontrol etmek isteyebilirsiniz.

* Tam Teşekküllü Çözümler

Doğrulama, ziyaret edilen alanları takip etme ve form teslimini içeren eksiksiz bir çözüm arıyorsanız Formik popüler seçeneklerden biridir. Bununla birlikte, aynı kontrol bileşenlerinin ve yönetim durumunun aynı ilkeleri üzerine kuruludur - bu yüzden bunları öğrenmeyi ihmal etmeyin.