

Props

Ustun Ozgur

2015-11-03 Tue

Props

- ▶ Short for properties
- ▶ Both native components like `div`, `p` and our custom components like `HelloWorld` can take props
- ▶ Key-value attributes delimited by equals sign
- ▶ To pass `{name: "ustun"}` to `HelloWorld` component
- ▶ `<HelloWorld name="Ustun"/>`

Props

- ▶ Can be accessed in the render method and other lifecycle methods
- ▶ Accessed as `this.props`
- ▶ name key from previous example: `this.props.name`

```
var HelloWorld = React.createClass({  
  render: function () {  
    return <div>{"Hello " + this.props.name}</div>;  
  });
```

A More Complex Example

- ▶ Aim: Greet a number of people
- ▶ People array: ["John", "Mary", "Susan"]

```
var People = React.createClass({  
  render: function () {  
    var people = ["John", "Mary", "Susan"];  
    var peopleNodes = [];  
  
    for (var i = 0; i < people.length; i++) {  
      peopleNodes.push(<HelloWorld name={people[i]}/>);  
    }  
    return <div>{peopleNodes}</div>;  
  });
```

A More Complex Example: Using Map

```
var People = React.createClass({  
  render: function () {  
    var people = ["John", "Mary", "Susan"];  
  
    var peopleNodes = people.map(function (person) {  
      return <HelloWorld name={person}/>;  
    });  
  
    return <div>{peopleNodes}</div>;  
  });
```

Cont'd

- ▶ We don't even need the intermediate variable
- ▶ Can directly embed in JSX expression

```
var People = React.createClass({  
  render: function () {  
    var people = ["John", "Mary", "Susan"];  
  
    return <div>  
      {people.map(function (person) {  
        return <HelloWorld name={person}/>; })}  
    </div>;  
  });
```

Exercise:

- ▶ Modify the todo list application so that a list of todo items (a list of strings) is passed as props.
- ▶ That is, the ToDos component should be passed a list such as:
- ▶ ["Buy Tomatoes", "Buy Potatoes"]
- ▶ Modify the list in code and ensure that the output is updated accordingly.
- ▶ Modify the render method so that the todo items are always sorted alphabetically.

Events and Functions as Props

- ▶ Any JavaScript expression can be passed down as props.
- ▶ Functions in JS are first class values: We can pass functions as props.
- ▶ For event handling, we have props like `onClick`, `onBlur`
- ▶ Pass functions for these props.

Attaching an onClick handler

```
var HelloWorld = React.createClass({
  onClick: function () {
    console.log("Hello " + this.props.name);
  },
  render: function () {
    return <div onClick={this.onClick}>
      {"Hello " + this.props.name}
    </div>;
  }
});
```

People List

- ▶ People list and HelloWorld child components
- ▶ Assume that the event handler logic is in People component, not HelloWorld.
- ▶ HelloWorld is just passed an onClick property (event handler)
- ▶ It executes whatever is passed from above.

```
var HelloWorld = React.createClass({  
  render: function () {  
    return <div onClick={this.props.onClick}>  
      {"Hello " + this.props.name}  
    </div>;  
  }  
});
```

Complexity in the Parent Component

```
var People = React.createClass({
  onClick: function (name) {
    console.log("Hello " + name);
  },
  render: function () {
    var people = ["John", "Mary", "Susan"];

    return <div>
      {people.map(function (person) {
        return <HelloWorld
          onClick={this.onClick.bind(this, person)}
          name={person}/>;
      }).bind(this))}
    </div>;
  });
```

What Changed?

- ▶ We modified onClick handler so that it accepts a name argument.

```
onClick: function (name) {
```

- ▶ We customized the onClick handler passed to each HelloWorld by binding the name parameter to the current person name.

```
onClick={this.onClick.bind(this, person)}
```

Bind method

- ▶ Introduced in ES5
- ▶ "creates a new function that, when called, has its `this` keyword set to the provided value, with a given sequence of arguments preceding any provided when the new function is called."
- ▶ First purpose: bind the `this` value
- ▶ Second purpose: bind the arguments: Create **partial** function

Example: Function that Adds 5

```
function add(a, b) { return a + b; }
```

- ▶ We want to fix a to 5.
- ▶ Value of `this` not important since unused
- ▶ Bind `this` to null

```
var add5 = add.bind(null, 5)  
console.log(add5(3)); // 8
```

Exercise:

- ▶ Think about how you can rename the method `console.log` to `l`.
- ▶ Does a simple `var l = console.log` work correctly? What is the correct solution?

Re-visit previous example

```
var People = React.createClass({
  onClick: function (name) {
    console.log("Hello " + name);
  },
  render: function () {
    var people = ["John", "Mary", "Susan"];
    return <div>
      {people.map(function (person) {
        return <HelloWorld
          onClick={this.onClick.bind(this, person)}
          name={person}/>;
      }).bind(this))}
    </div>;
  });
```

- ▶ Second bind: Binding this value.
- ▶ First bind: Binding this value and the additional argument

Verbose Version

```
var People = React.createClass({
  onClick: function (name) {
    console.log("Hello " + name);
  },
  render: function () {
    var people = ["John", "Mary", "Susan"];

    return <div>
      {people.map(function (person) {
        var boundFunction = this.onClick.bind(this, person);
        return <HelloWorld
          onClick={boundFunction}
          name={person}/>;
      }).bind(this))}
    </div>;
  });
```

Another alternative

- ▶ Store the current this value in a variable, for example that

```
var People = React.createClass({
  onClick: function (name) {
    console.log("Hello " + name);
  },
  render: function () {
    var people = ["John", "Mary", "Susan"];
    var that = this;
    return <div>
      {people.map(function (person) {
        return <HelloWorld
          onClick={that.onClick.bind(that, person)}
          name={person}/>;
      })}
    </div>;
  });
```

Alternative using `_.partial`

```
var People = React.createClass({
  onClick: function (name) {
    console.log("Hello " + name);
  },
  render: function () {
    var people = ["John", "Mary", "Susan"];
    var that = this;
    return <div>
      {people.map(function (person) {
        return <HelloWorld
          onClick={_.partial(that.onClick, person)}
          name={person}/>;
      })}
    </div>;
  });
```

getDefaultProps

- ▶ A component method like render
- ▶ Default property values

Example: HelloWorld with Default Greeting

```
var HelloWorld = React.createClass({
  getDefaultProps: function () {
    return {greeting: 'Hello'}
  },
  render: function () {
    return <div>
      {this.props.greeting} {this.props.name}
    </div>;
  }
});
```

```
var People = React.createClass({
  render: function () {
    return <div>
      <HelloWorld name="John"/>
      <HelloWorld greeting="Hola" name="Mary"/>
    </div>}});
```