

Lab 7

Tracing react-post

Datalab

WeatherMood

Today Forecast

Search

📍 Hsinchu

Today: Broken Clouds

31°c

?

Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的



DataLab.

Main

Search

📍 Hsinchu

Today: Broken Clouds

31°c

?

Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的



DataLab.

Main

Search

📍 Hsinchu

Today: Broken Clouds

Today

31°c

?

Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的



DataLab.

Main

Search

📍 Hsinchu

Today: Broken Clouds

Today

31° c

ⓘ Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的



DataLab.

PostForm

Main

Search

📍 Hsinchu

Today: Broken Clouds

31° c

Today

?] Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的

**PostForm****PostList**

DataLab.

WeatherMood

Today Forecast

Main

Search

WeatherFrom

Hsinchu

Today

Today: Broken Clouds

31° c

Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的



PostForm

PostList

DataLab.

WeatherMood Today Forecast

Main

Search

WeatherFrom

Hsinchu

31° c

Today: Broken Clouds

Today

Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的

PostItem



PostForm

PostList

DataLab.

📍 Hsinchu

Today: Broken Clouds

31° c

ⓘ Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的

PostItem



PostForm

PostList

DataLab.

WeatherMood

Today Forecast

Search

📍 Hsinchu

Today: Broken Clouds

31° c

ⓘ Mood ▾

What's on your mind?

Post

PostForm



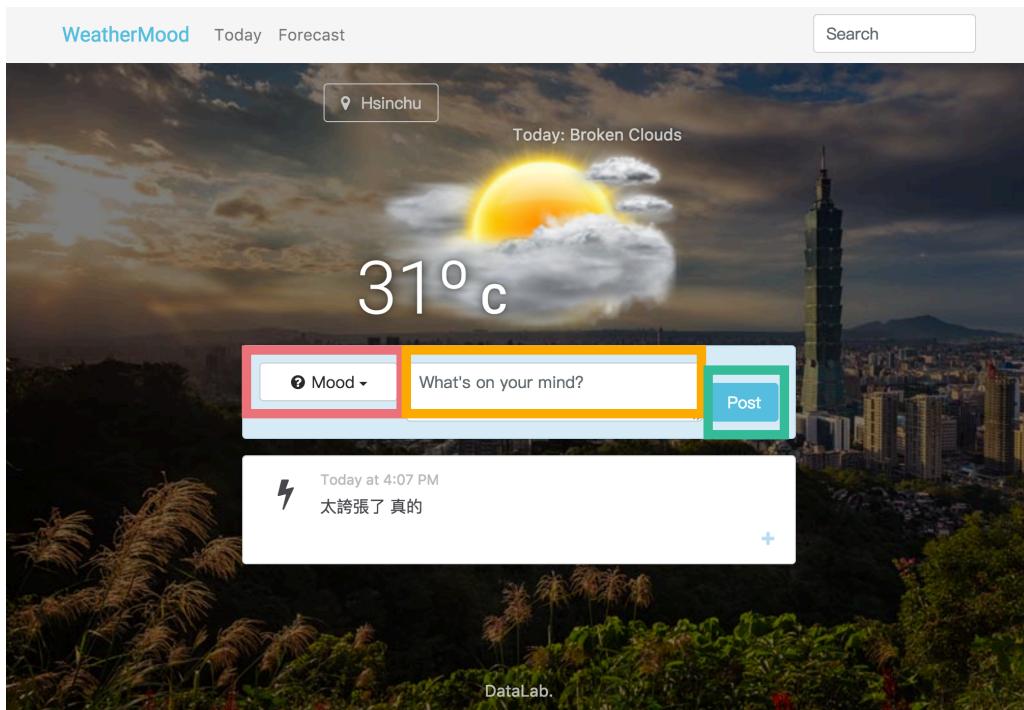
Today at 4:07 PM

太誇張了 真的



DataLab.

PostForm



```
return (
  <div className='post-form'>
    <Alert color='info' className={'d-flex flex-column flex-sm-row justify-content-center ${inputDangerous ? "border border-danger" : ""}'>
      <div className='mood align-self-start'>
        <ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
          <DropdownToggle className='mood-toggle' type='button' caret color="secondary">
            <i className={getMoodIcon(mood)}></i>&nbsp;{
              mood === 'na' ? 'Mood' : mood
            }
          </DropdownToggle>
          <DropdownMenu>
            <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Clear')}><i
              className='fa fa-times' /> Clear
            </DropdownItem>
            <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Clouds')}><i
              className='fa fa-sun-o' /> Clouds
            </DropdownItem>
            <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Drizzle')}><i
              className='fa fa-cloud' /> Drizzle
            </DropdownItem>
            <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Rain')}><i
              className='fa fa-cloud雨' /> Rain
            </DropdownItem>
            <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Thunder')}><i
              className='fa fa-thunderstorm' /> Thunder
            </DropdownItem>
            <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Snow')}><i
              className='fa fa-snowflake' /> Snow
            </DropdownItem>
            <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Windy')}><i
              className='fa fa-wind' /> Windy
            </DropdownItem>
          </DropdownMenu>
        </ButtonDropdown>
      </div>
      <Input className='input' type='textarea' getRef={el => {this.inputEl = el}} value={this.state
        .content}>
        <Button className='btn-post align-self-end' color="info" onClick={this.handlePost}>Post</Butt
      </Input>
    </Alert>
  </div>
);
```

PostForm - Dropdown

```
handleDropdownSelect(mood) {  
    this.setState({mood: mood});  
}
```

PostForm - Dropdown

```
handleDropdownSelect(mood) {  
    this.setState({mood: mood});  
}
```



PostForm - Dropdown

```
handleDropdownSelect(mood) {
    this.setState({mood: mood});
}
```



```
render() {
    const {inputValue, moodToggle, mood} = this.state;
    const inputDanger = this.state.inputDanger ? 'has-danger' : '';

    return (
        <div className='post-form'>
            <Alert color='info' className='d-flex flex-column flex-
                <div className='mood align-self-start'>
                    <ButtonDropdown type='button' isOpen={moodToggle}>
                        <DropdownToggle className='mood-toggle' type='button'>
                            <i className={getMoodIcon(mood)}></i>&nbsp;
                            mood === 'na' ? 'Mood' : mood
                        </DropdownToggle>
                    <ButtonDropdown.Menu>
                        <div>
                            <span>Action 1</span>
                            <span>Action 2</span>
                        </div>
                    </ButtonDropdown.Menu>
                </div>
            </Alert>
        </div>
    );
}
```

PostForm - Dropdown

```
handleDropdownSelect(mood) {
    this.setState({mood: mood});
}
```

```
render() {
    const {inputValue, moodToggle, mood} = this.state;
    const inputDanger = this.state.inputDanger ? 'has-danger' : '';

    return (
        <div className='post-form'>
            <Alert color='info' className='d-flex flex-column flex-nowrap align-items-center'>
                <div className='mood align-self-start'>
                    <ButtonDropdown type='button' isOpen={moodToggle}>
                        <DropdownToggle className='mood-toggle' type='button'>
                            <i className={getMoodIcon(mood)}></i>&nbsp;
                            {mood === 'na' ? 'Mood' : mood}
                        </DropdownToggle>
                    </ButtonDropdown>
                </div>
            </Alert>
        </div>
    );
}
```

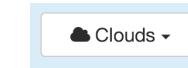
PostForm - Dropdown

```
handleDropdownSelect(mood) {
    this.setState({mood: mood});
}
```



```
render() {
    const {inputValue, moodToggle, mood} = this.state;
    const inputDanger = this.state.inputDanger ? 'has-danger' : '';

    return (
        <div className='post-form'>
            <Alert color='info' className={`d-flex flex-column flex-`}>
                <div className='mood align-self-start'>
                    <ButtonDropdown type='button' isOpen={moodToggle}>
                        <DropdownToggle className='mood-toggle' type='button'>
                            <i className={getMoodIcon(mood)}></i>&nbsp;
                            {mood === 'na' ? 'Mood' : mood}
                        </DropdownToggle>
                    </ButtonDropdown>
                </div>
            </Alert>
        </div>
    );
}
```



PostForm - Input

```
handleInputChange(e) {
  const text = e.target.value
  this.setState({inputValue: text});
  if (text) {
    this.setState({inputDanger: false});
  }
}
```

PostForm - Input

```
handleInputChange(e) {
  const text = e.target.value
  this.setState({inputValue: text});
  if (text) {
    this.setState({inputDanger: false});
  }
}
```



PostForm - Input

```
handleInputChange(e) {  
  const text = e.target.value  
  this.setState({inputValue: text});  
  if (text) {  
    this.setState({inputDanger: false});  
  }  
}
```



render()

```
  value={this.state.inputValue}  
  lePost>Post</Button>
```

PostForm - Input

```
handleInputChange(e) {  
  const text = e.target.value  
  this.setState({inputValue: text});  
  if (text) {  
    this.setState({inputDanger: false});  
  }  
}
```

render()

```
  value={this.state.inputValue}  
  lePost>Post</Button>
```

PostForm - Input

```
handleInputChange(e) {  
  const text = e.target.value;  
  this.setState({inputValue: text});  
  if (text) {  
    this.setState({inputDanger: false});  
  }  
}
```

render()

```
  value={this.state.inputValue}  
  lePost>Post</Button>
```

太誇張了 真的

PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color="secondary">
    <i className={getMoodIcon(mood)}></i>&ampnbsp{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



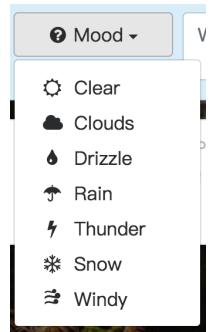
PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```

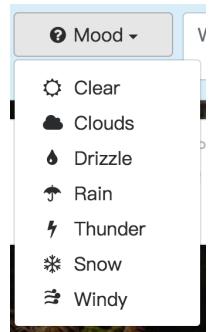


PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



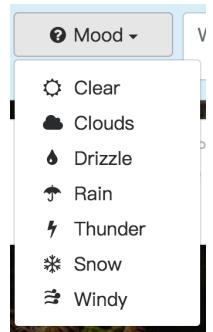
PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



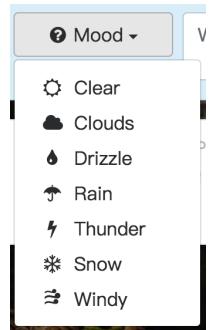
PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';
return (
  <div className='post-form'>
    <Alert color='info' className={`d-flex flex-column flex-sm-row justify-content-center ${inputDanger}`}>
```

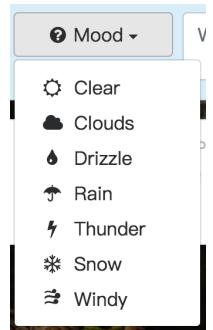
PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';
return (
  <div className='post-form'>
    <Alert color='info' className={`d-flex flex-column flex-sm-row justify-content-center ${inputDanger}`}>
```

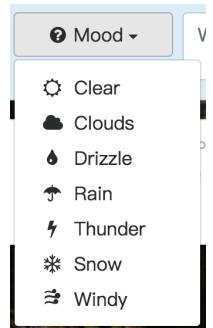
PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';
return (
  <div className='post-form'>
    <Alert color='info' className={`d-flex flex-column flex-sm-row justify-content-center ${inputDanger}`}>
```

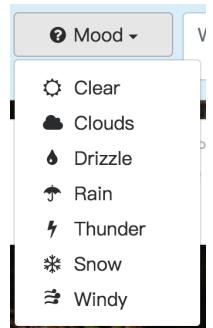
PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';
return (
  <div className='post-form'>
    <Alert color='info' className={`d-flex flex-column flex-sm-row justify-content-center ${inputDanger}`}>
```

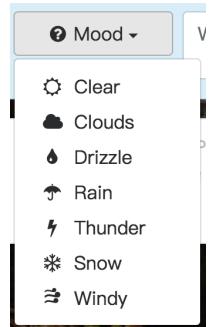
PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }

  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

```
<ButtonDropdown type='button' isOpen={moodToggle} toggle={this.handleMoodToggle}>
  <DropdownToggle className='mood-toggle' type='button' caret color='secondary'>
    <i className={getMoodIcon(mood)}></i>&nbsp;{
      mood === 'na' ? 'Mood' : mood
    }
  </DropdownToggle>
</ButtonDropdown>
```



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';
return (
  <div className='post-form'>
    <Alert color='info' className={`d-flex flex-column flex-sm-row justify-content-center ${inputDanger}`}>
```

What's on your mind?

PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```

PostForm - Post

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



PostForm - Post

Today.jsx

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



PostForm - Post

Today.jsx

```
handlePost() {  
  if (this.state.mood === 'na') {  
    this.setState({moodToggle: true});  
    return;  
  }  
  if (!this.state.inputValue) {  
    this.setState({inputDanger: true});  
    return;  
  }  
  
  this.props.onPost(this.state.mood, this.state.inputValue);  
  this.setState({  
    inputValue: '',  
    mood: 'na'  
  });  
}
```



```
<div className='posts'>  
  <PostForm onPost={this.handleCreatePost} />  
  <PostList posts={posts} onVote={this.handleCreateVote} />{  
    postLoading &&  
    <Alert color='warning' className='loading'>Loading...</Alert>  
  }  
</div>
```

PostForm - Post

Today.jsx

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote} />{
    postLoading &&
      <Alert color='warning' className='loading'>Loading...</Alert>
  }
</div>
```

PostForm - Post

Today.jsx

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote} />{
    postLoading &&
      <Alert color='warning' className='loading'>Loading...</Alert>
  }
</div>
```



PostForm - Post

Today.jsx

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote} />{
    postLoading &&
      <Alert color='warning' className='loading'>Loading...</Alert>
  }
</div>
```



```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

PostForm - Post

Today.jsx

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote} />{
    postLoading &&
      <Alert color='warning' className='loading'>Loading...</Alert>
  }
</div>
```



```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```



PostForm - Post

Today.jsx

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote} />{
    postLoading &&
      <Alert color='warning' className='loading'>Loading...</Alert>
  }
</div>
```



```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```



```
import WeatherDisplay from 'components/WeatherDisplay.jsx';
import WeatherForm from 'components/WeatherForm.jsx';
import PostForm from 'components/PostForm.jsx';
import PostList from 'components/PostList.jsx';
import {getWeather, cancelWeather} from 'api/open-weather-map.js';
import {listPosts, createPost, createVote} from 'api/posts.js';
```

PostForm - Post

Today.jsx

```
handlePost() {
  if (this.state.mood === 'na') {
    this.setState({moodToggle: true});
    return;
  }
  if (!this.state.inputValue) {
    this.setState({inputDanger: true});
    return;
  }

  this.props.onPost(this.state.mood, this.state.inputValue);
  this.setState({
    inputValue: '',
    mood: 'na'
  });
}
```



```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote} />{
    postLoading &&
      <Alert color='warning' className='loading'>Loading...</Alert>
  }
</div>
```



```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```



```
import WeatherDisplay from 'components/WeatherDisplay.jsx';
import WeatherForm from 'components/WeatherForm.jsx';
import PostForm from 'components/PostForm.jsx';
import PostList from 'components/PostList.jsx';
import {getWeather, cancelWeather} from 'api/open-weather-map.js';
import {listPosts, createPost, createVote} from 'api/posts.js';
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```



createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```



```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

createPost

Today.jsx

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```



createPost

Today.jsx

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

createPost

Today.jsx

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

createPost

Today.jsx

api/posts.js

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}

export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

Today.jsx

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

Today.jsx

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

```
listPosts(searchText) {
  this.setState({
    postLoading: true
  }, () => {
    listPosts(searchText).then(posts => {
      this.setState({
        posts,
        postLoading: false
      });
    }).catch(err => {
      console.error('Error listing posts', err);

      this.setState({
        posts: [],
        postLoading: false
      });
    });
  });
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

Today.jsx

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

```
listPosts(searchText) {
  this.setState({
    postLoading: true
  }, () => {
    listPosts(searchText).then(posts => {
      this.setState({
        posts,
        postLoading: false
      });
    }).catch(err => {
      console.error('Error listing posts', err);

      this.setState({
        posts: [],
        postLoading: false
      });
    });
  });
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

Today.jsx

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

```
listPosts(searchText) {
  this.setState({
    postLoading: true
  }, () => {
    listPosts(searchText).then(posts => {
      this.setState({
        posts,
        postLoading: false
      });
    }).catch(err => {
      console.error('Error listing posts', err);

      this.setState({
        posts: [],
        postLoading: false
      });
    });
  });
}
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

Today.jsx

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

```
listPosts(searchText) {
  this.setState({
    postLoading: true
  }, () => {
    listPosts(searchText).then(posts => {
      this.setState({
        posts,
        postLoading: false
      });
    }).catch(err => {
      console.error('Error listing posts', err);

      this.setState({
        posts: [],
        postLoading: false
      });
    });
  });
}
```

```
import {listPosts, createPost, createVote} from 'api/posts.js';
```

createPost

api/posts.js

```
export function createPost(mood, text) {
  return new Promise((resolve, reject) => {
    resolve(_createPost(mood, text));
  });
}
```

```
// Simulated server-side code
function _createPost(mood, text) {
  const newPost = {
    id: uuid(),
    mood: mood,
    text: text,
    ts: moment().unix(),
    clearVotes: 0,
    cloudsVotes: 0,
    drizzleVotes: 0,
    rainVotes: 0,
    thunderVotes: 0,
    snowVotes: 0,
    windyVotes: 0
  };
  const posts = [
    newPost,
    ..._listPosts()
  ];
  localStorage.setItem(postKey, JSON.stringify(posts));
  return newPost;
}
```

Today.jsx

```
handleCreatePost(mood, text) {
  createPost(mood, text).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating posts', err);
  });
}
```

```
listPosts(searchText) {
  this.setState({
    postLoading: true
  }, () => {
    listPosts(searchText).then(posts => {
      this.setState({
        posts,
        postLoading: false
      });
    }).catch(err => {
      console.error('Error listing posts', err);
    });
    this.setState({
      posts: [],
      postLoading: false
    });
  });
}
```

```
import {listPosts, createPost, createVote} from 'api/posts.js'
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```



listPosts

api/posts.js

```
export function listPosts(searchText = '') {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve(_listPosts(searchText));
        }, 500);
    });
}
```

→

```
// Simulated server-side code
function _listPosts(searchText = '') {
    let postString = localStorage.getItem(postKey);
    let posts = postString ? JSON.parse(postString) : [];
    if (posts.length > 0 && searchText) {
        posts = posts.filter(p => {
            return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
        });
    }
    return posts;
};
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve(_listPosts(searchText));
        }, 500);
    });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
    let postString = localStorage.getItem(postKey);
    let posts = postString ? JSON.parse(postString) : [];
    if (posts.length > 0 && searchText) {
        posts = posts.filter(p => {
            return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
        });
    }
    return posts;
};
```



listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

```
listPosts(searchText) {
  this.setState({
    postLoading: true
}, () => {
  listPosts(searchText).then(posts => {
    this.setState({
      posts,
      postLoading: false
    });
  }).catch(err => {
    console.error('Error listing posts', err);

    this.setState({
      posts: [],
      postLoading: false
    });
  });
});
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

```
listPosts(searchText) {
  this.setState({
    postLoading: true
}, () => {
  listPosts(searchText).then(posts => {
    this.setState({
      posts,
      postLoading: false
    });
  }).catch(err => {
    console.error('Error listing posts', err);

    this.setState({
      posts: [],
      postLoading: false
    });
  });
});
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

```
listPosts(searchText) {
  this.setState({
    postLoading: true
}, () => {
  listPosts(searchText).then(posts => {
    this.setState({
      posts,
      postLoading: false
    });
  }).catch(err => {
    console.error('Error listing posts', err);

    this.setState({
      posts: [],
      postLoading: false
    });
  });
});
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

```
listPosts(searchText) {
  this.setState({
    postLoading: true
}, () => {
  listPosts(searchText).then(posts => {
    this.setState({
      posts,
      postLoading: false
    });
  }).catch(err => {
    console.error('Error listing posts', err);

    this.setState({
      posts: [],
      postLoading: false
    });
  });
});
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

```
listPosts(searchText) {
  this.setState({
    postLoading: true
}, () => {
  listPosts(searchText).then(posts => {
    this.setState({
      posts,
      postLoading: false
    });
  }).catch(err => {
    console.error('Error listing posts', err);

    this.setState({
      posts: [],
      postLoading: false
    });
  });
});
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

```
listPosts(searchText) {
  this.setState({
    postLoading: true
}, () => {
  listPosts(searchText).then(posts => {
    this.setState({
      posts,
      postLoading: false
    });
  }).catch(err => {
    console.error('Error listing posts', err);

    this.setState({
      posts: [],
      postLoading: false
    });
  });
});
```

render()

```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote}>{(
    postLoading &&
    <Alert color='warning' className='loading'>Loading...</Alert>
  )}
</div>
```

listPosts

api/posts.js

```
export function listPosts(searchText = '') {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(_listPosts(searchText));
    }, 500);
  });
}
```

```
// Simulated server-side code
function _listPosts(searchText = '') {
  let postString = localStorage.getItem(postKey);
  let posts = postString ? JSON.parse(postString) : [];
  if (posts.length > 0 && searchText) {
    posts = posts.filter(p => {
      return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase()) !== -1
    });
  }
  return posts;
};
```

Today.jsx

```
listPosts(searchText) {
  this.setState({
    postLoading: true
}, () => {
  listPosts(searchText).then(posts => {
    this.setState({
      posts,
      postLoading: false
    });
  }).catch(err => {
    console.error('Error listing posts', err);
    this.setState({
      posts: [],
      postLoading: false
    });
  });
});
```

render()

```
<div className='posts'>
  <PostForm onPost={this.handleCreatePost} />
  <PostList posts={posts} onVote={this.handleCreateVote}>{(
    postLoading &&
    <Alert color='warning' className='loading'>Loading...</Alert>
  )}
</div>
```

📍 Hsinchu

Today: Broken Clouds

31°c

?

Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的



PostList

DataLab.

PostList

```
render() {
  const {posts} = this.props;

  let children = (
    <ListGroupItem className='empty d-flex justify-content-center align-items-center'>
      <div className='empty-text'>No post here.<br />Go add some posts.</div>
    </ListGroupItem>
  );
  if (posts.length) {
    children = posts.map(p => (
      <ListGroupItem key={p.id} action>
        <PostItem {...p} onVote={this.handleVote} />
      </ListGroupItem>
    )));
  }

  return (
    <div className='post-list'>
      <ListGroup>{children}</ListGroup>
    </div>
  );
}
```

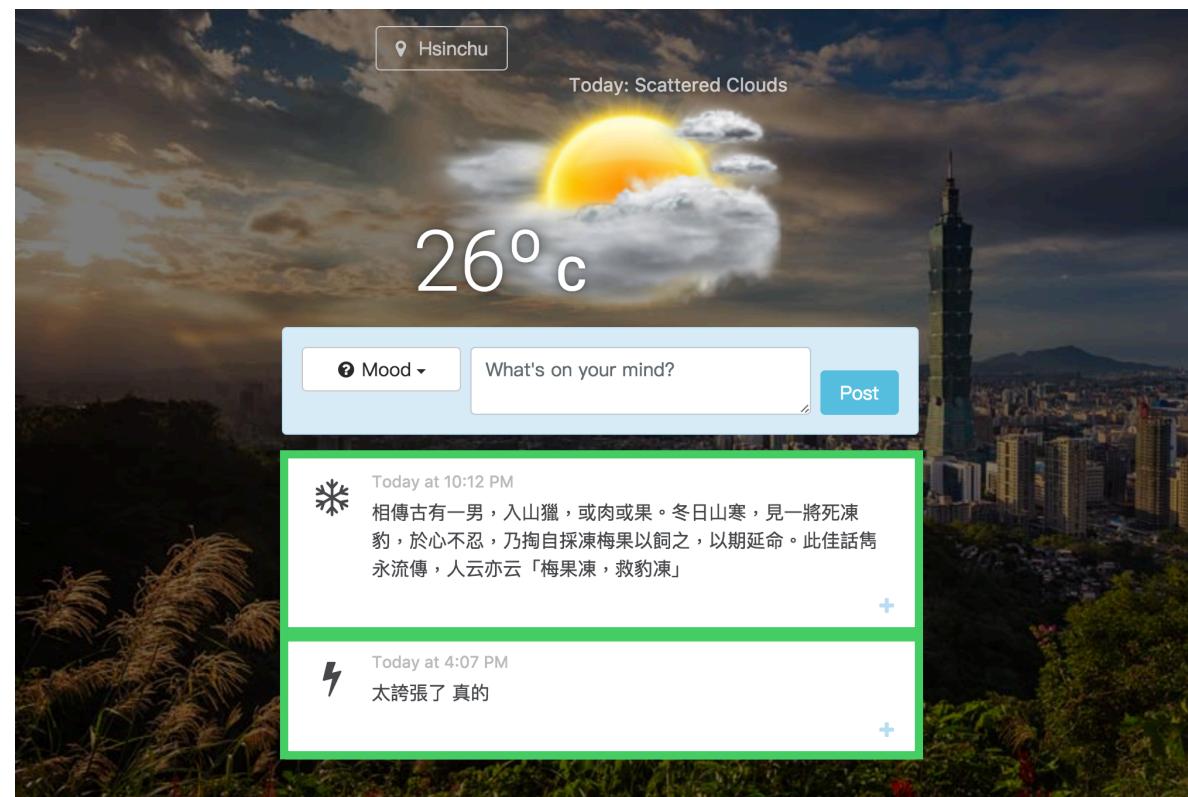


PostList

```
render() {
  const {posts} = this.props;

  let children = (
    <ListGroupItem className='empty d-flex justify-content-center align-items-center'>
      <div className='empty-text'>No post here.<br />Go add some posts.</div>
    </ListGroupItem>
  );
  if (posts.length) {
    children = posts.map(p => (
      <ListGroupItem key={p.id} action>
        <PostItem {...p} onVote={this.handleVote} />
      </ListGroupItem>
    )));
  }

  return (
    <div className='post-list'>
      <ListGroup>{children}</ListGroup>
    </div>
  );
}
```

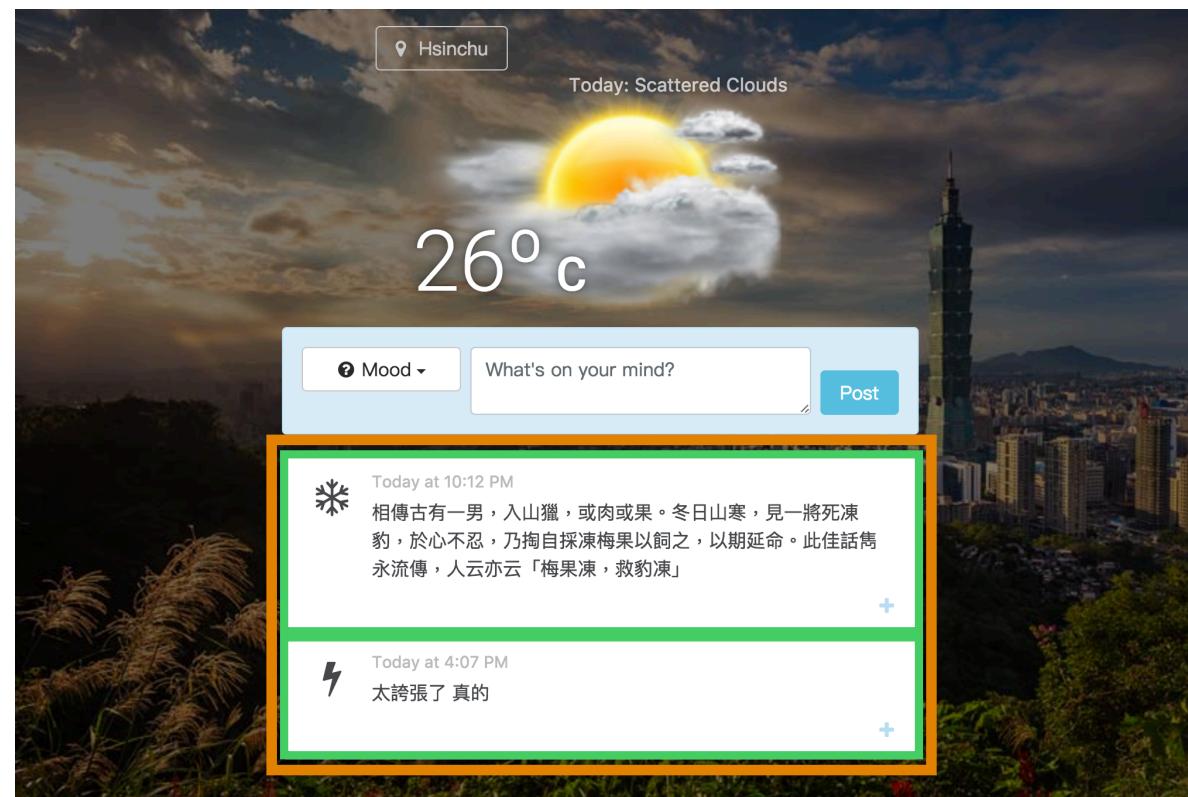


PostList

```
render() {
  const {posts} = this.props;

  let children = (
    <ListGroupItem className='empty d-flex justify-content-center align-items-center'>
      <div className='empty-text'>No post here.<br />Go add some posts.</div>
    </ListGroupItem>
  );
  if (posts.length) {
    children = posts.map(p => (
      <ListGroupItem key={p.id} action>
        <PostItem {...p} onVote={this.handleVote} />
      </ListGroupItem>
    ));
  }

  return (
    <div className='post-list'>
      <ListGroup>{children}</ListGroup>
    </div>
  );
}
```



WeatherMood

Today Forecast

Search

📍 Hsinchu

Today: Broken Clouds

31° c

?

Mood ▾

What's on your mind?

Post



Today at 4:07 PM

太誇張了 真的

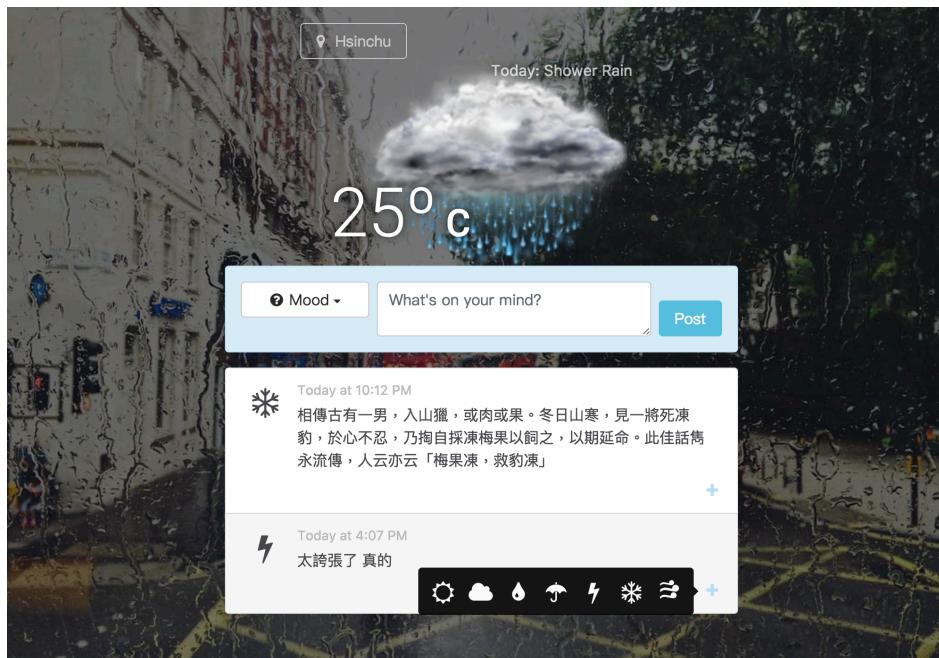
PostItem



DataLab.

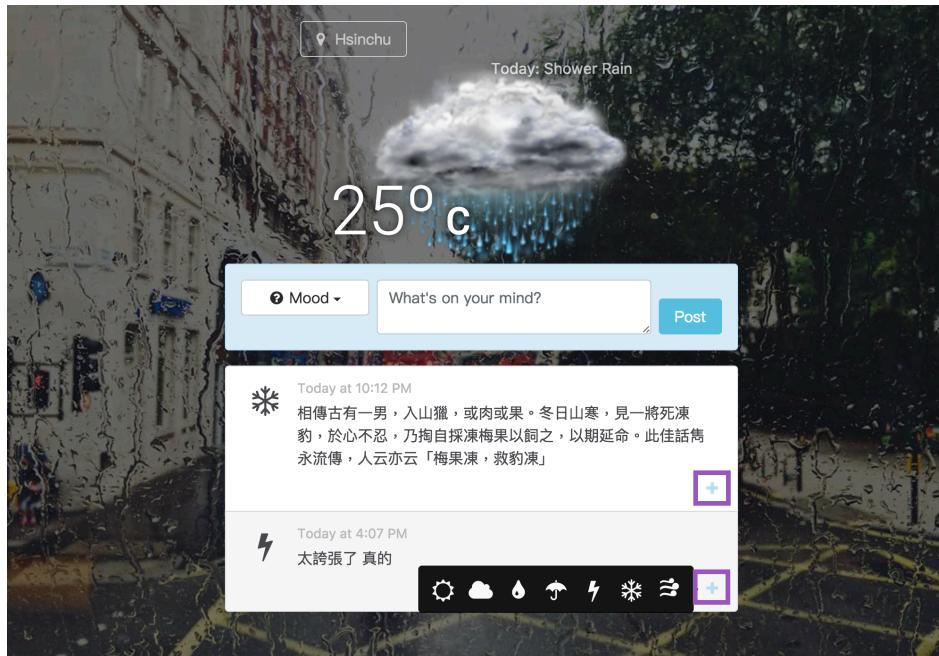
PostItem

```
<Tooltip placement='left' isOpen={tooltipOpen} autohide={false} target={`post-item-vote-${id}`} toggle={this.handleTooltipToggle}>
  <i className={'vote-tooltip ${getMoodIcon('Clear')}' } onClick={() => this.handleVote('Clear')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Clouds')}' } onClick={() => this.handleVote('Clouds')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Drizzle')}' } onClick={() => this.handleVote('Drizzle')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Rain')}' } onClick={() => this.handleVote('Rain')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Thunder')}' } onClick={() => this.handleVote('Thunder')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Snow')}' } onClick={() => this.handleVote('Snow')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Windy')}' } onClick={() => this.handleVote('Windy')}></i>
</Tooltip>
```



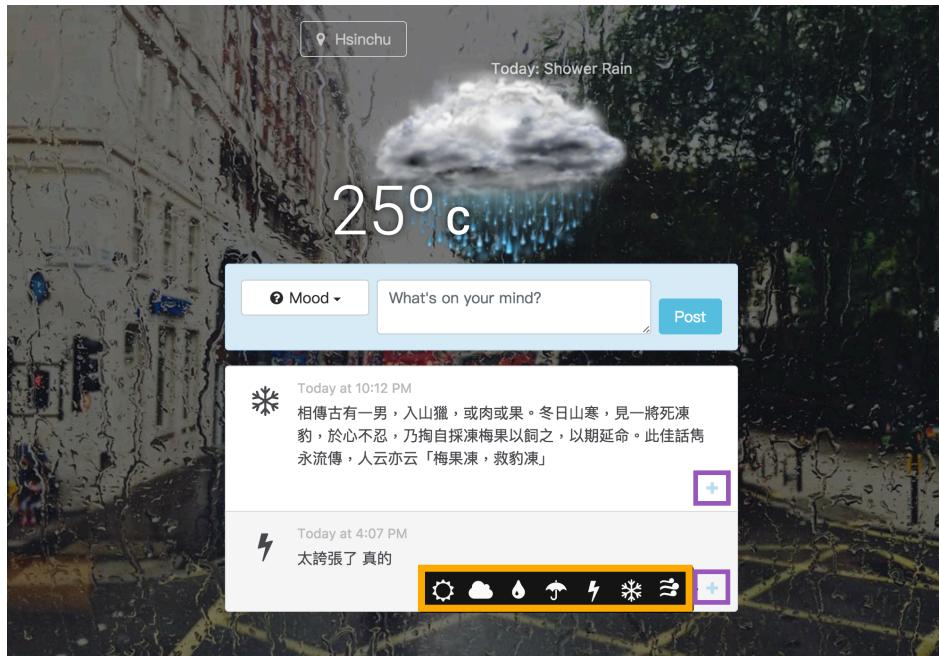
PostItem

```
<Tooltip placement='left' isOpen={tooltipOpen} autohide={false} target={`post-item-vote-${id}`} toggle={this.handleTooltipToggle}>
  <i className={'vote-tooltip ${getMoodIcon('Clear')}' } onClick={() => this.handleVote('Clear')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Clouds')}' } onClick={() => this.handleVote('Clouds')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Drizzle')}' } onClick={() => this.handleVote('Drizzle')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Rain')}' } onClick={() => this.handleVote('Rain')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Thunder')}' } onClick={() => this.handleVote('Thunder')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Snow')}' } onClick={() => this.handleVote('Snow')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Windy')}' } onClick={() => this.handleVote('Windy')}></i>
</Tooltip>
```



PostItem

```
<Tooltip placement='left' isOpen={tooltipOpen} autohide={false} target={`post-item-vote-${id}`} toggle={this.handleTooltipToggle}>
  <i className={'vote-tooltip ${getMoodIcon('Clear')}'} onClick={() => this.handleVote('Clear')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Clouds')}'} onClick={() => this.handleVote('Clouds')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Drizzle')}'} onClick={() => this.handleVote('Drizzle')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Rain')}'} onClick={() => this.handleVote('Rain')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Thunder')}'} onClick={() => this.handleVote('Thunder')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Snow')}'} onClick={() => this.handleVote('Snow')}></i>&nbsp;
  <i className={'vote-tooltip ${getMoodIcon('Windy')}'} onClick={() => this.handleVote('Windy')}></i>
</Tooltip>
```



PostItem

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```

PostItem

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
 });
}
```



PostItem

render()

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
});
}
```



```
<div className='vote d-flex justify-content-end'>
  <div className='vote-results'>
    {clearVotes > 0 && (<span><i className={getMoodIcon('Clear')}></i>&nbsp;{clearVotes}&nbsp;</span>)}
    {cloudsVotes > 0 && (<span><i className={getMoodIcon('Clouds')}></i>&nbsp;{cloudsVotes}&nbsp;&nbsp;</span>)}
    {drizzleVotes > 0 && (<span><i className={getMoodIcon('Drizzle')}></i>&nbsp;{drizzleVotes}&nbsp;&nbsp;</span>)}
    {rainVotes > 0 && (<span><i className={getMoodIcon('Rain')}></i>&nbsp;{rainVotes}&nbsp;&nbsp;</span>)}
    {thunderVotes > 0 && (<span><i className={getMoodIcon('Thunder')}></i>&nbsp;{thunderVotes}&nbsp;&nbsp;</span>)}
    {snowVotes > 0 && (<span><i className={getMoodIcon('Snow')}></i>&nbsp;{snowVotes}&nbsp;&nbsp;</span>)}
    {windyVotes > 0 && (<span><i className={getMoodIcon('Windy')}></i>&nbsp;{windyVotes}&nbsp;&nbsp;</span>)}
  </div>
  <div className='vote-plus'>
    <i id={'post-item-vote-${id}'} className='fa fa-plus'></i>
  </div>
</div>
```

PostItem

render()

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
});
}
```



```
<div className='vote d-flex justify-content-end'>
  <div className='vote-results'>
    {clearVotes > 0 && (<span><i className={getMoodIcon('Clear')}></i>&nbsp;{clearVotes}&nbsp;</span>)}
    {cloudsVotes > 0 && (<span><i className={getMoodIcon('Clouds')}></i>&nbsp;{cloudsVotes}&nbsp;&nbsp;</span>)}
    {drizzleVotes > 0 && (<span><i className={getMoodIcon('Drizzle')}></i>&nbsp;{drizzleVotes}&nbsp;&nbsp;</span>)}
    {rainVotes > 0 && (<span><i className={getMoodIcon('Rain')}></i>&nbsp;{rainVotes}&nbsp;&nbsp;</span>)}
    {thunderVotes > 0 && (<span><i className={getMoodIcon('Thunder')}></i>&nbsp;{thunderVotes}&nbsp;&nbsp;</span>)}
    {snowVotes > 0 && (<span><i className={getMoodIcon('Snow')}></i>&nbsp;{snowVotes}&nbsp;&nbsp;</span>)}
    {windyVotes > 0 && (<span><i className={getMoodIcon('Windy')}></i>&nbsp;{windyVotes}&nbsp;&nbsp;</span>)}
  </div>
  <div className='vote-plus'>
    <i id={'post-item-vote-${id}'} className='fa fa-plus'></i>
  </div>
</div>
```

PostItem

render()

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```



```
<div className='vote d-flex justify-content-end'>
  <div className='vote-results'>
    {clearVotes > 0 && (<span><i className={getMoodIcon('Clear')}></i>&nbsp;{clearVotes}&nbsp;</span>)}
    {cloudsVotes > 0 && (<span><i className={getMoodIcon('Clouds')}></i>&nbsp;{cloudsVotes}&nbsp;&nbsp;</span>)}
    {drizzleVotes > 0 && (<span><i className={getMoodIcon('Drizzle')}></i>&nbsp;{drizzleVotes}&nbsp;&nbsp;</span>)}
    {rainVotes > 0 && (<span><i className={getMoodIcon('Rain')}></i>&nbsp;{rainVotes}&nbsp;&nbsp;</span>)}
    {thunderVotes > 0 && (<span><i className={getMoodIcon('Thunder')}></i>&nbsp;{thunderVotes}&nbsp;&nbsp;</span>)}
    {snowVotes > 0 && (<span><i className={getMoodIcon('Snow')}></i>&nbsp;{snowVotes}&nbsp;&nbsp;</span>)}
    {windyVotes > 0 && (<span><i className={getMoodIcon('Windy')}></i>&nbsp;{windyVotes}&nbsp;&nbsp;</span>)}
  </div>
  <div className='vote-plus'>
    <i id={'post-item-vote-${id}'} className='fa fa-plus'></i>
  </div>
</div>
```



PostItem

render()

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```



```
<div className='vote d-flex justify-content-end'>
  <div className='vote-results'>
    {clearVotes > 0 && (<span><i className={getMoodIcon('Clear')}></i>&nbsp;{clearVotes}&nbsp;</span>)}
    {cloudsVotes > 0 && (<span><i className={getMoodIcon('Clouds')}></i>&nbsp;{cloudsVotes}&nbsp;</span>)}
    {drizzleVotes > 0 && (<span><i className={getMoodIcon('Drizzle')}></i>&nbsp;{drizzleVotes}&nbsp;</span>)}
    {rainVotes > 0 && (<span><i className={getMoodIcon('Rain')}></i>&nbsp;{rainVotes}&nbsp;</span>)}
    {thunderVotes > 0 && (<span><i className={getMoodIcon('Thunder')}></i>&nbsp;{thunderVotes}&nbsp;</span>)}
    {snowVotes > 0 && (<span><i className={getMoodIcon('Snow')}></i>&nbsp;{snowVotes}&nbsp;</span>)}
    {windyVotes > 0 && (<span><i className={getMoodIcon('Windy')}></i>&nbsp;{windyVotes}&nbsp;</span>)}
  </div>
  <div className='vote-plus'>
    <i id={'post-item-vote-${id}'} className='fa fa-plus'></i>
  </div>
</div>
```



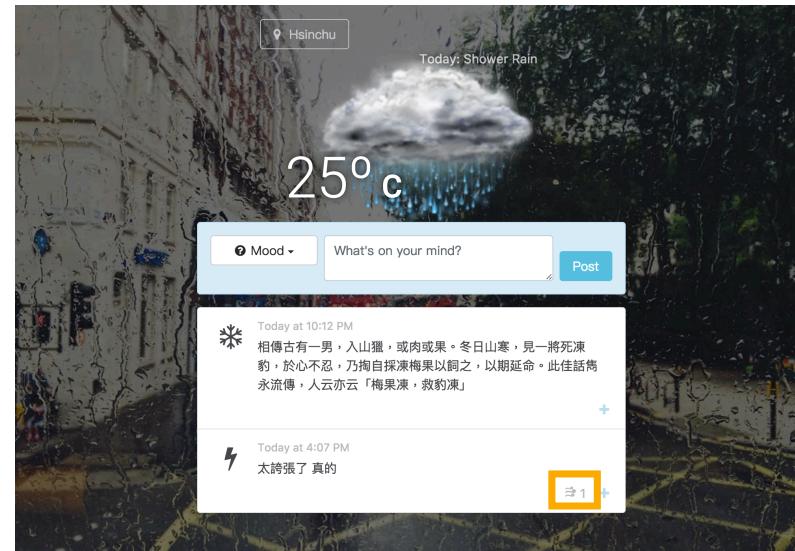
PostItem

render()

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```



```
<div className='vote d-flex justify-content-end'>
  <div className='vote-results'>
    {clearVotes > 0 && (<span><i className={getMoodIcon('Clear')}></i>&nbsp;{clearVotes}&nbsp;</span>)}
    {cloudsVotes > 0 && (<span><i className={getMoodIcon('Clouds')}></i>&nbsp;{cloudsVotes}&nbsp;</span>)}
    {drizzleVotes > 0 && (<span><i className={getMoodIcon('Drizzle')}></i>&nbsp;{drizzleVotes}&nbsp;</span>)}
    {rainVotes > 0 && (<span><i className={getMoodIcon('Rain')}></i>&nbsp;{rainVotes}&nbsp;</span>)}
    {thunderVotes > 0 && (<span><i className={getMoodIcon('Thunder')}></i>&nbsp;{thunderVotes}&nbsp;</span>)}
    {snowVotes > 0 && (<span><i className={getMoodIcon('Snow')}></i>&nbsp;{snowVotes}&nbsp;</span>)}
    {windyVotes > 0 && (<span><i className={getMoodIcon('Windy')}></i>&nbsp;{windyVotes}&nbsp;</span>)}
  </div>
  <div className='vote-plus'>
    <i id={'post-item-vote-${id}'} className='fa fa-plus'></i>
  </div>
</div>
```



handleVote

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```

handleVote

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```

handleVote

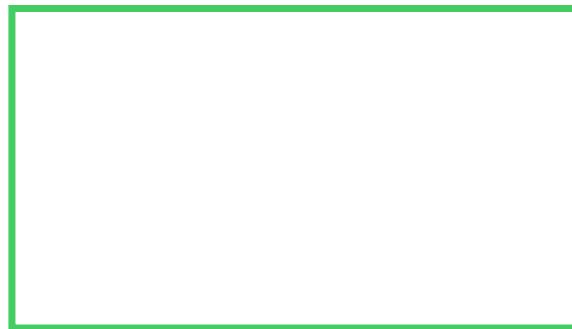
```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```



handleVote

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



handleVote

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



render()

```
<ListGroupItem key={p.id} action>  
  <PostItem {...p} onVote={this.handleVote} />  
</ListGroupItem>
```

handleVote

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



render()

```
<ListGroupItem key={p.id} action>  
  <PostItem {...p} onVote={this.handleVote} />  
</ListGroupItem>
```

handleVote

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



render()

```
<ListGroupItem key={p.id} action>  
  <PostItem {...p} onVote={this.handleVote} />  
</ListGroupItem>  
  
handleVote(id, mood) {  
  this.props.onVote(id, mood);  
}
```

handleVote

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



render()

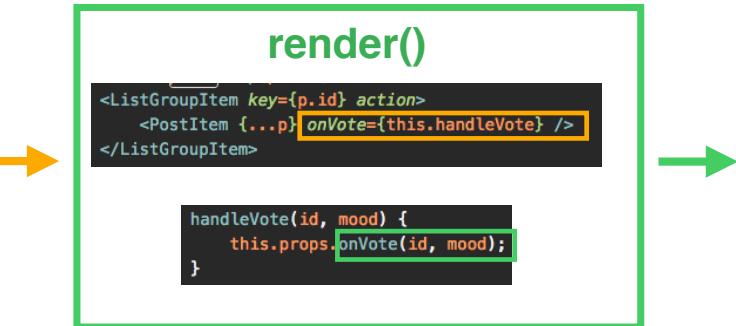
```
<ListGroupItem key={p.id} action>  
  <PostItem {...p} onVote={this.handleVote} />  
</ListGroupItem>
```

```
handleVote(id, mood) {  
  this.props.onVote(id, mood);  
}
```

handleVote

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



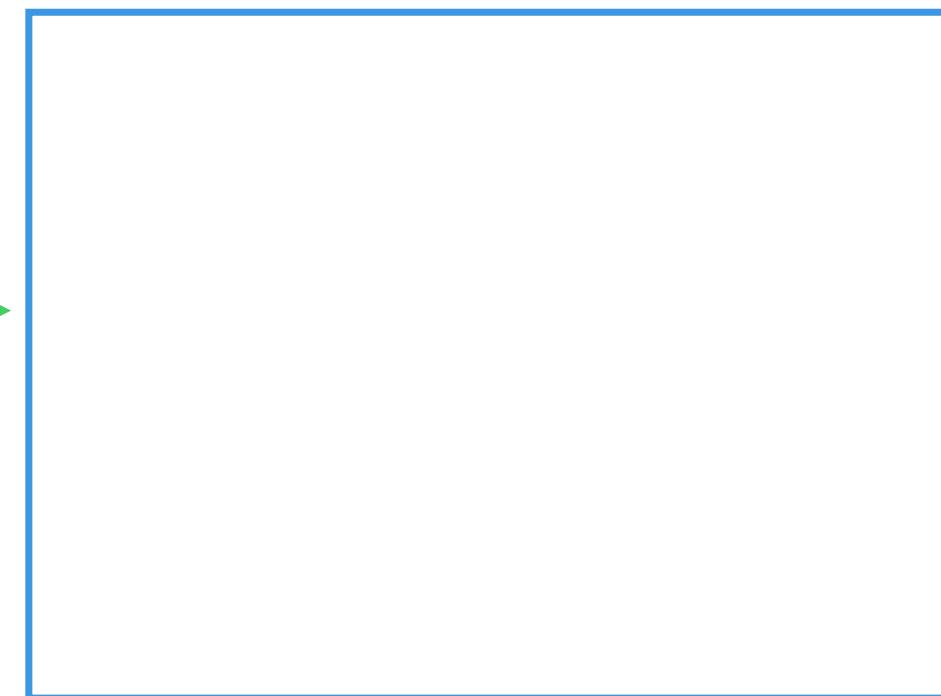
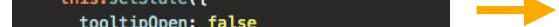
handleVote

Today.jsx

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```

```
render() {  
  return (  
    <ListGroupItem key={p.id} action>  
      <PostItem {...p} onVote={this.handleVote} />  
    </ListGroupItem>  
  
    handleVote(id, mood) {  
      this.props.onVote(id, mood);  
    }  
  );  
}
```



handleVote

Today.jsx

PostList.jsx

render()

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



```
<ListGroupItem key={p.id} action>  
  <PostItem {...p} onVote={this.handleVote} />  
</ListGroupItem>  
  
handleVote(id, mood) {  
  this.props.onVote(id, mood);  
}
```



```
<PostList posts={posts} onVote={this.handleCreateVote} />  
  postLoading &&  
  <Alert color='warning' className='loading'>Loading...</Alert>  
}
```

handleVote

Today.jsx

PostList.jsx

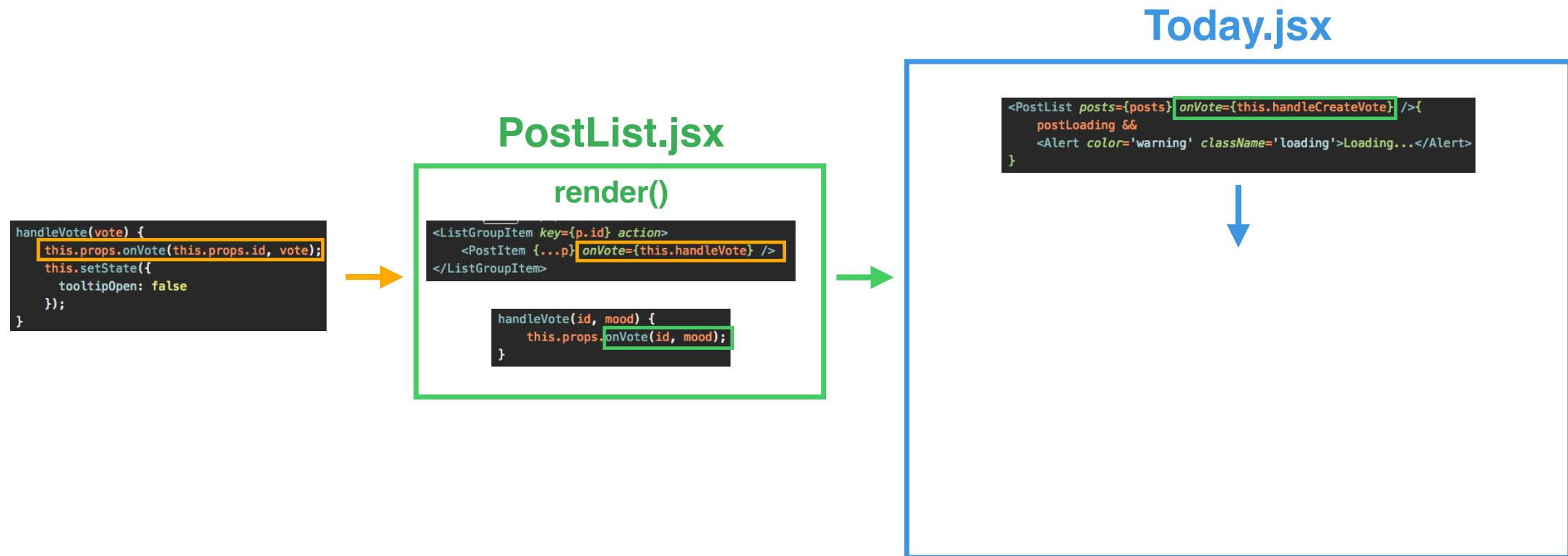
render()

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```

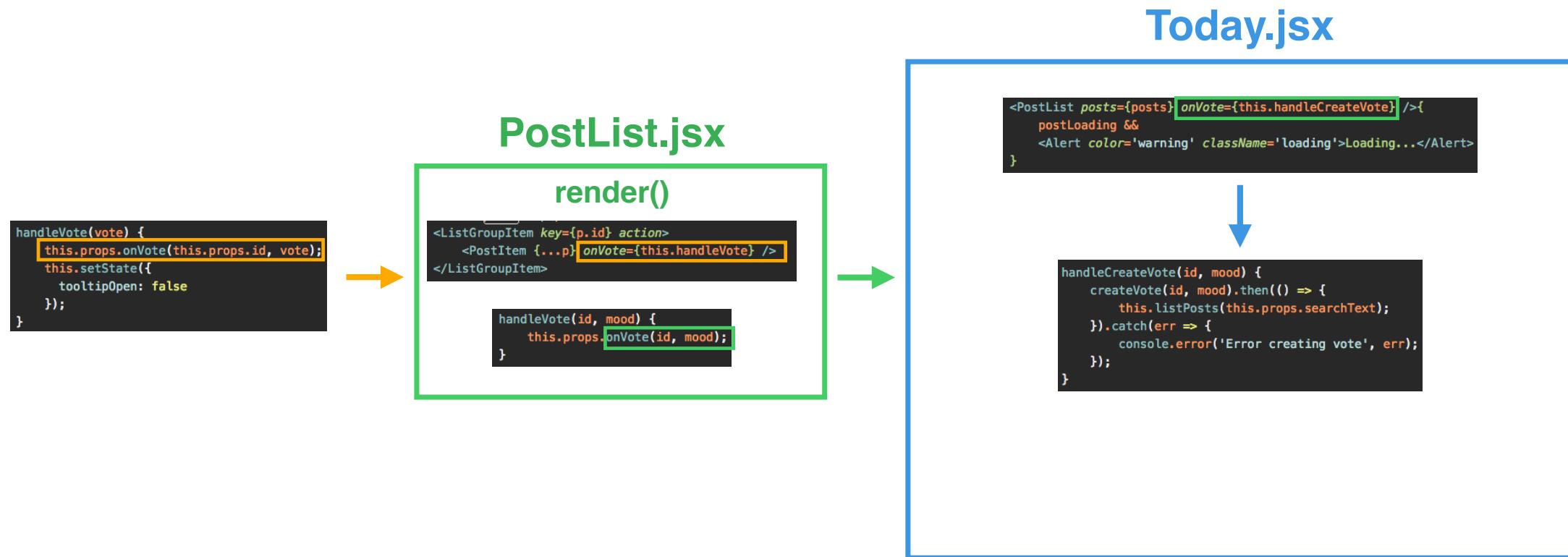
```
<ListGroupItem key={p.id} action>  
  <PostItem {...p} onVote={this.handleVote} />  
</ListGroupItem>  
  
handleVote(id, mood) {  
  this.props.onVote(id, mood);  
}
```

```
<PostList posts={posts} onVote={this.handleCreateVote} />{  
  postLoading &&  
  <Alert color='warning' className='loading'>Loading...</Alert>  
}
```

handleVote



handleVote



handleVote

Today.jsx

PostList.jsx

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```



render()

```
<ListGroupItem key={p.id} action>
  <PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>

handleVote(id, mood) {
  this.props.onVote(id, mood);
}
```



```
<PostList posts={posts} onVote={this.handleCreateVote} />{
  postLoading &&
  <Alert color='warning' className='loading'>Loading...</Alert>
}
```



```
handleCreateVote(id, mood) {
  createVote(id, mood).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating vote', err);
  });
}
```

handleVote

Today.jsx

PostList.jsx

```
handleVote(vote) {  
  this.props.onVote(this.props.id, vote);  
  this.setState({  
    tooltipOpen: false  
  });  
}
```



render()

```
<ListGroupItem key={p.id} action>  
  <PostItem {...p} onVote={this.handleVote} />  
</ListGroupItem>  
  
handleVote(id, mood) {  
  this.props.onVote(id, mood);  
}
```



```
<PostList posts={posts} onVote={this.handleCreateVote} />{  
  postLoading &&  
  <Alert color='warning' className='loading'>Loading...</Alert>  
}
```



```
handleCreateVote(id, mood) {  
  createVote(id, mood).then(() => {  
    this.listPosts(this.props.searchText);  
  }).catch(err => {  
    console.error('Error creating vote', err);  
  });  
}
```



handleVote

Today.jsx

PostList.jsx

```
handleVote(vote) {
  this.props.onVote(this.props.id, vote);
  this.setState({
    tooltipOpen: false
  });
}
```



render()

```
<ListGroupItem key={p.id} action>
  <PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>

handleVote(id, mood) {
  this.props.onVote(id, mood);
}
```



```
<PostList posts={posts} onVote={this.handleCreateVote} />{
  postLoading &&
  <Alert color='warning' className='loading'>Loading...</Alert>
}
```



```
handleCreateVote(id, mood) {
  createVote(id, mood).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating vote', err);
  });
}
```



```
import {listPosts, createPost, createVote} from 'api/posts.js';
```

handleVote

api/posts.js

```
export function createVote(id, mood) {
  return new Promise((resolve, reject) => {
    _createVote(id, mood);
    resolve();
  });
}

// Simulated server-side code
function _createVote(id, mood) {
  const posts = _listPosts().map(p => {
    if (p.id === id) {
      p[mood.toLowerCase() + 'Votes']++;
    }
    return p;
  });
  localStorage.setItem(postKey, JSON.stringify(posts));
}
```

handleVote

api/posts.js

```
export function createVote(id, mood) {
  return new Promise((resolve, reject) => {
    _createVote(id, mood);
    resolve();
  });
}

// Simulated server-side code
function _createVote(id, mood) {
  const posts = _listPosts().map(p => {
    if (p.id === id) {
      p[mood.toLowerCase() + 'Votes']++;
    }
    return p;
  });
  localStorage.setItem(postKey, JSON.stringify(posts));
}
```



handleVote

api/posts.js

```
export function createVote(id, mood) {
  return new Promise((resolve, reject) => {
    _createVote(id, mood);
    resolve();
  });
}

// Simulated server-side code
function _createVote(id, mood) {
  const posts = _listPosts().map(p => {
    if (p.id === id) {
      p[mood.toLowerCase() + 'Votes']++;
    }
    return p;
  });
  localStorage.setItem(postKey, JSON.stringify(posts));
}
```

Today.jsx



handleVote

api/posts.js

```
export function createVote(id, mood) {
  return new Promise((resolve, reject) => {
    _createVote(id, mood);
    resolve();
  });
}

// Simulated server-side code
function _createVote(id, mood) {
  const posts = _listPosts().map(p => {
    if (p.id === id) {
      p[mood.toLowerCase() + 'Votes']++;
    }
    return p;
  });
  localStorage.setItem(postKey, JSON.stringify(posts));
}
```

Today.jsx

```
handleCreateVote(id, mood) {
  createVote(id, mood).then(() => {
    this.listPosts(this.props.searchText);
  }).catch(err => {
    console.error('Error creating vote', err);
  });
}
```

handleVote

api/posts.js

```
export function createVote(id, mood) {
  return new Promise((resolve, reject) => {
    _createVote(id, mood);
    resolve();
  });
}

// Simulated server-side code
function _createVote(id, mood) {
  const posts = _listPosts().map(p => {
    if (p.id === id) {
      p[mood.toLowerCase() + 'Votes']++;
    }
    return p;
  });
  localStorage.setItem(postKey, JSON.stringify(posts));
}
```

Today.jsx

```
handleCreateVote(id, mood) {
  createVote(id, mood).then(() =>
    this.listPosts(this.props.searchText);
  ).catch(err => {
    console.error('Error creating vote', err);
  });
}
```

Today's Mission

- A user can react to a post **only once**
- Hint : create Vote objects which contains userID and postID
- You should also handle vote counts in Post objects when a user's reaction to a post changes
- Deadline : 4/18 23:59