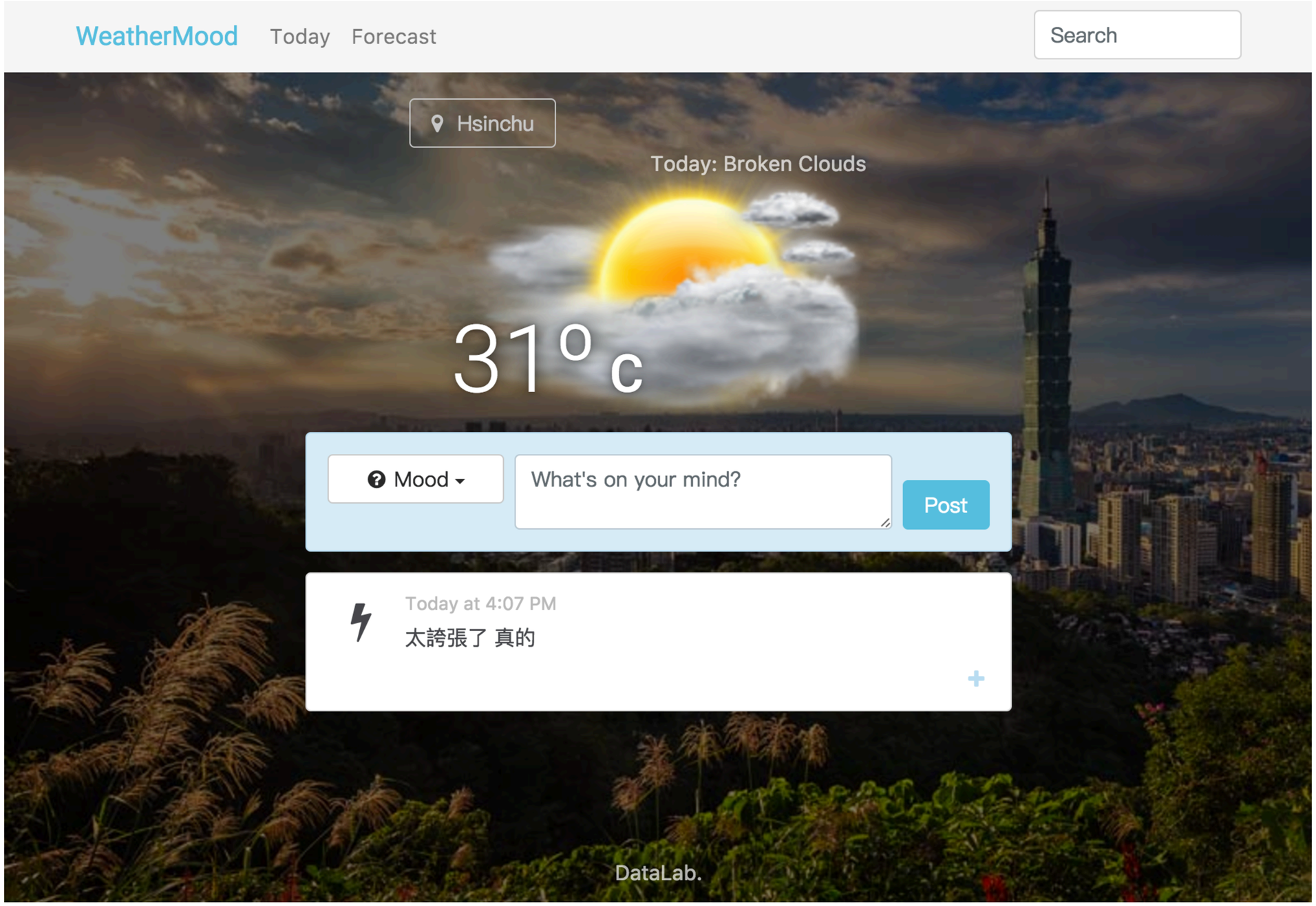
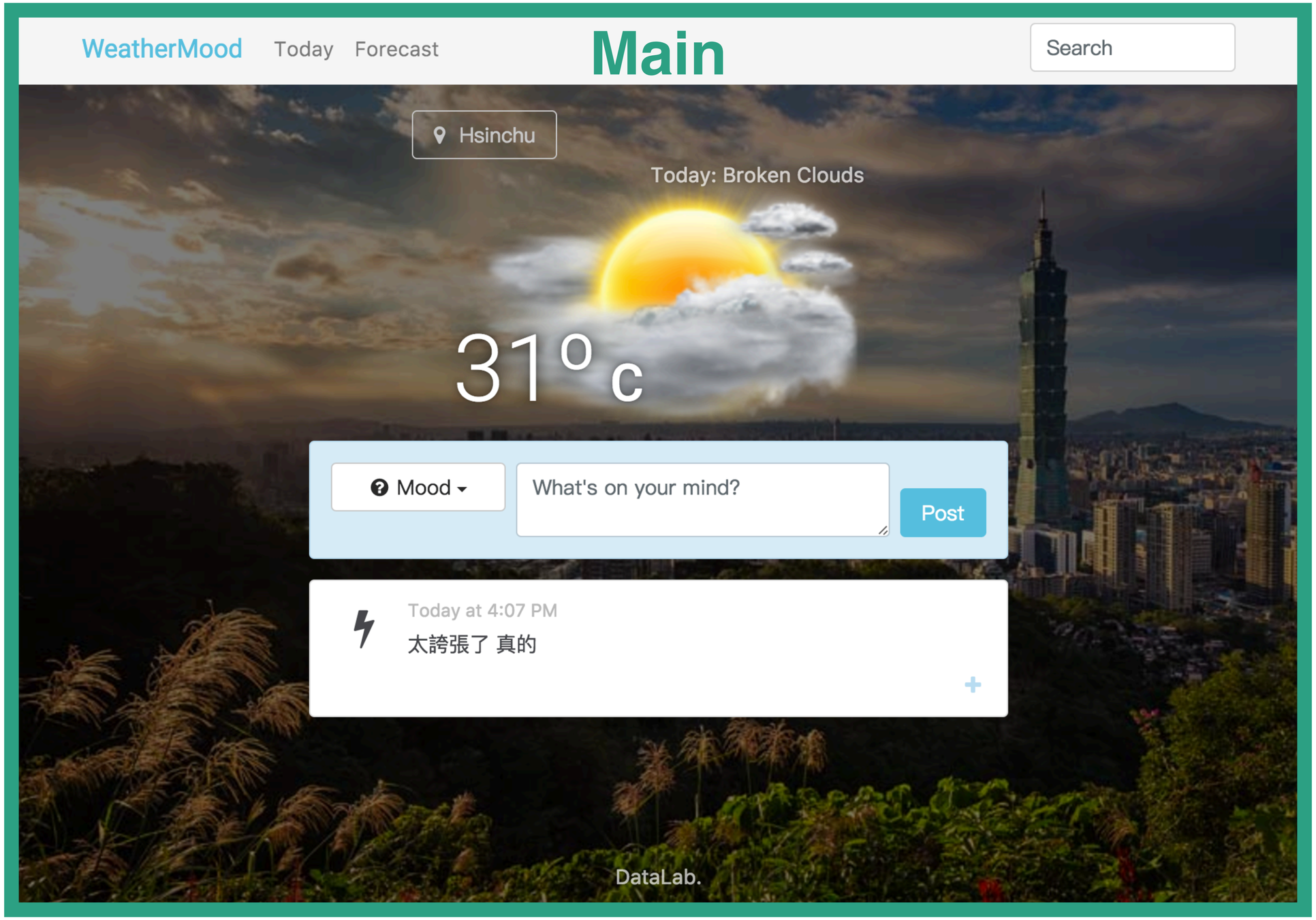
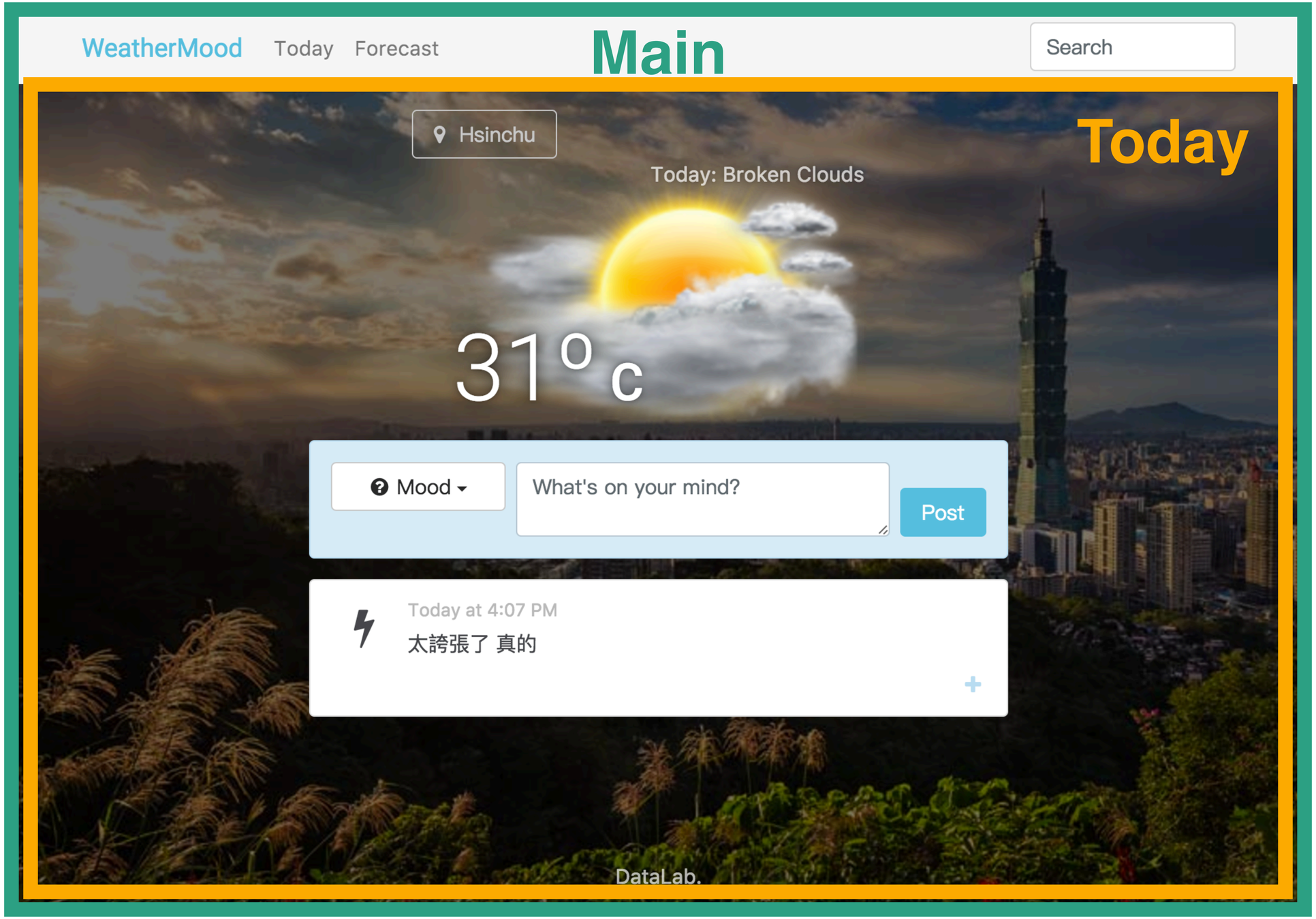
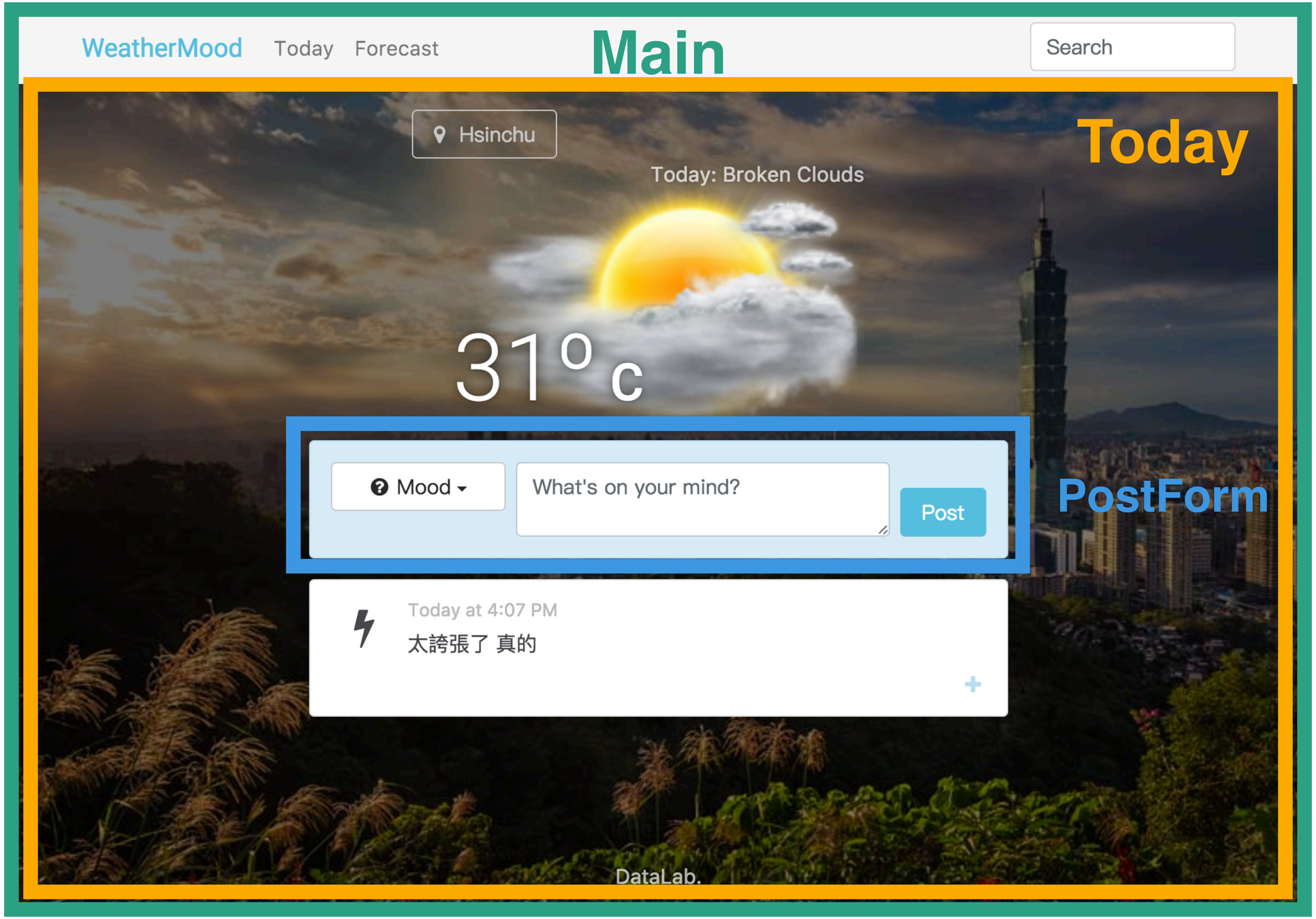
Lab 7
Tracing react-post

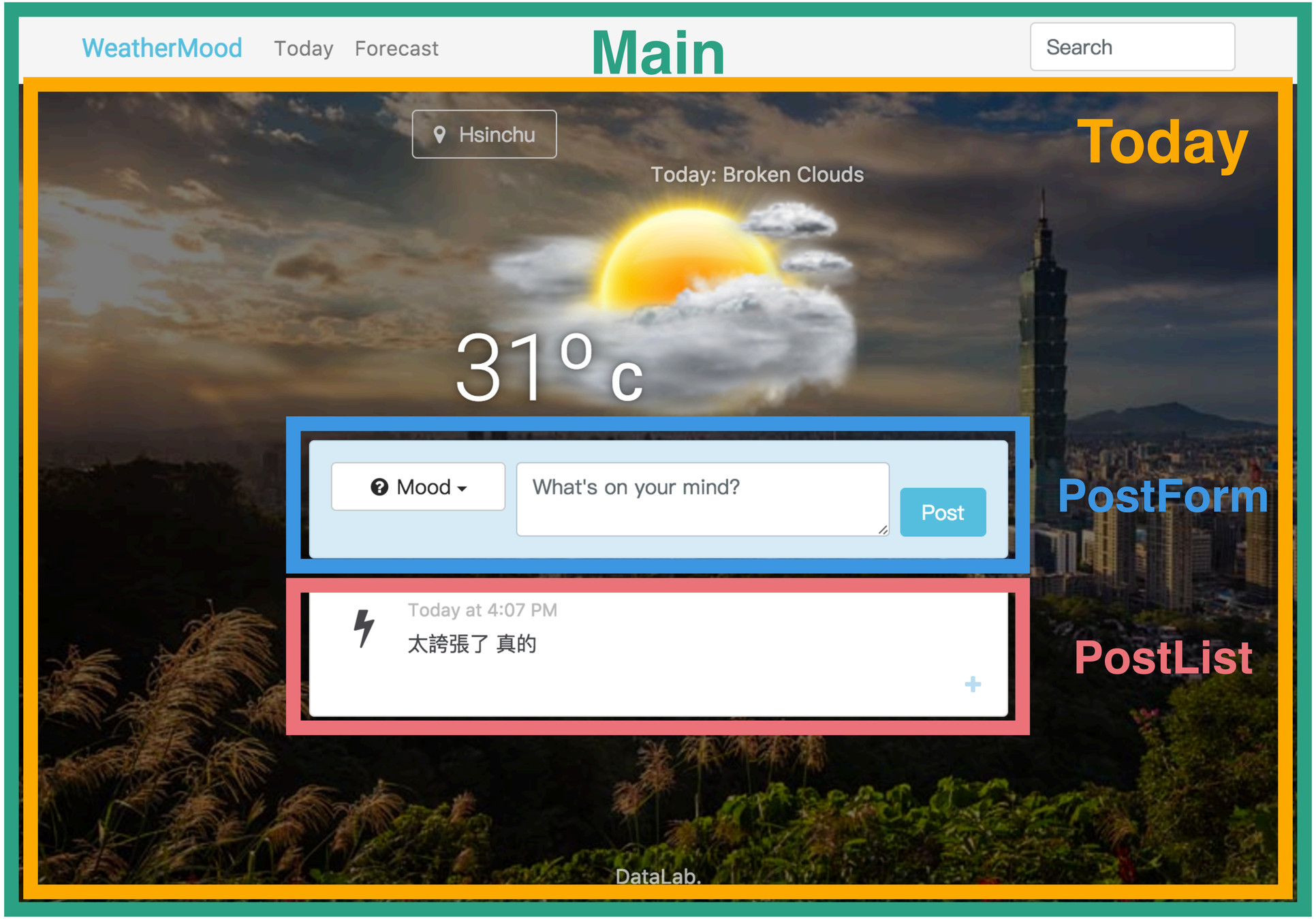
Datalab

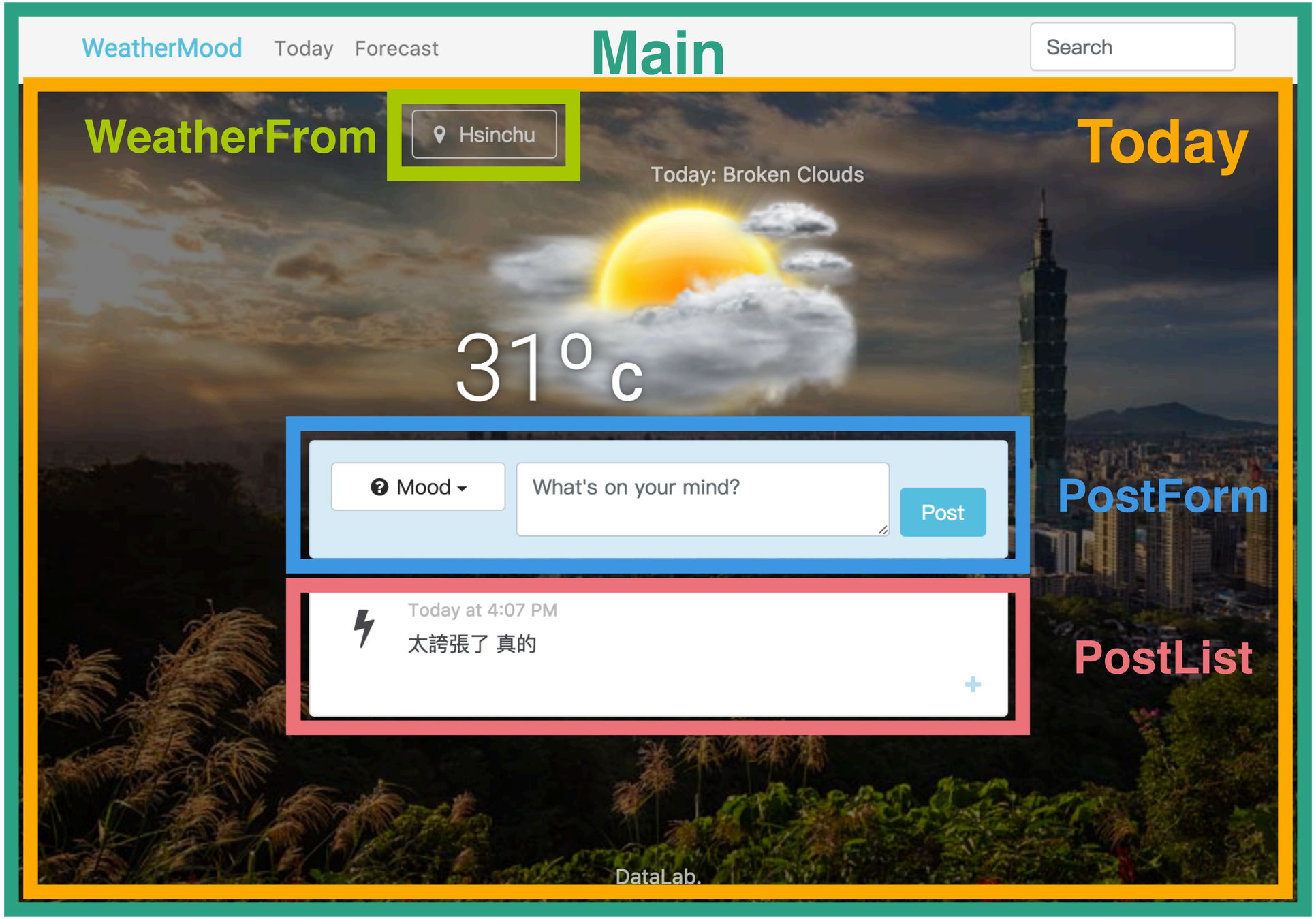

WeatherMood
Today
Forecast
Search
Hsinchu
Today: Broken Clouds
31° c
Mood
What's on your mind?
Post
Today at 4:07 PM
太誇張了 真的
DataLab.

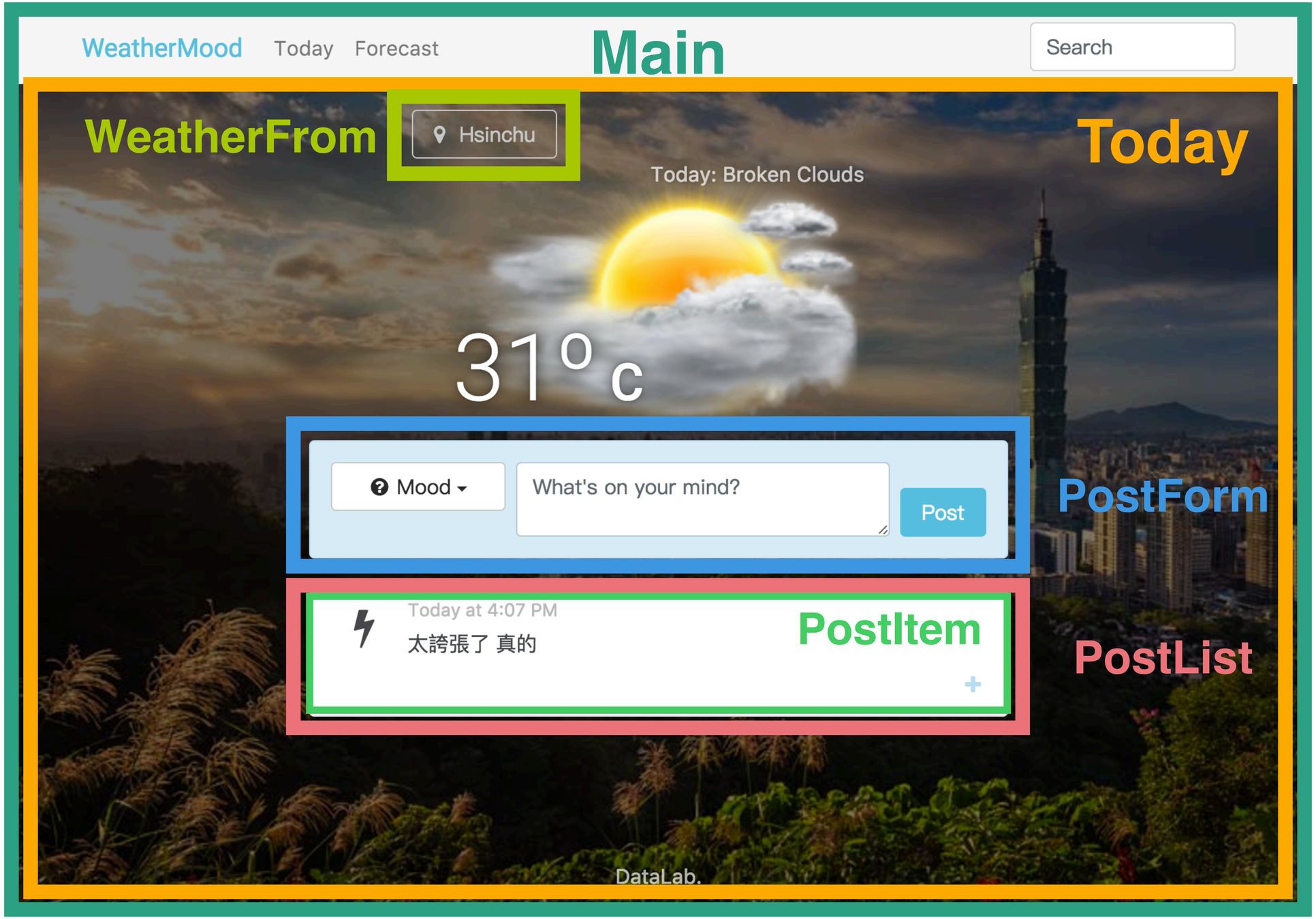

WeatherMood
Today
Forecast
Main
Search
Hsinchu
Today: Broken Clouds
31°c
Mood
What's on your mind?
Post
Today at 4:07 PM
太誇張了 真的
DataLab.

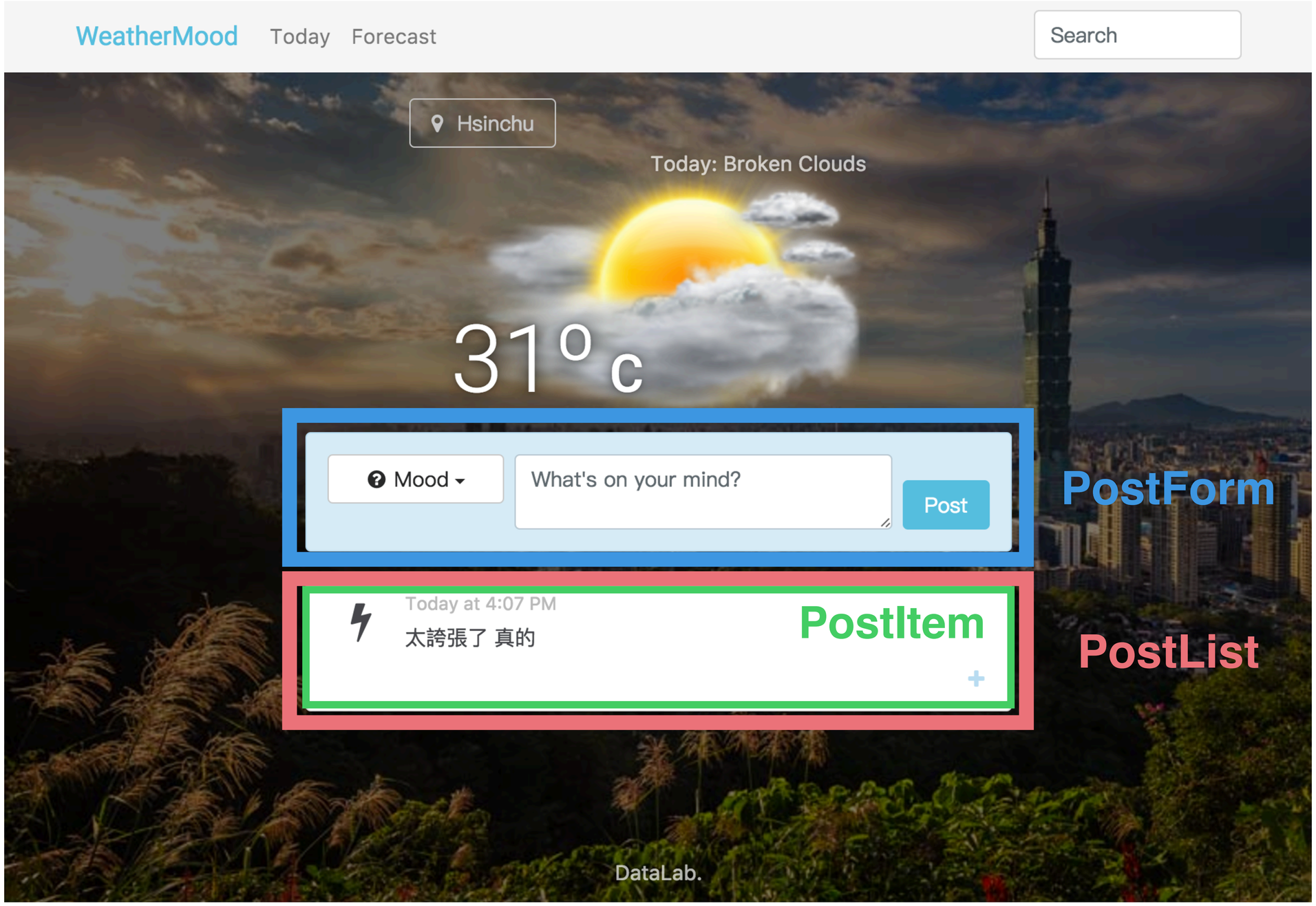

WeatherMood
Today
Forecast
Main
Search
Hsinchu
Today
Today: Broken Clouds
31° c
Mood
What's on your mind?
Post
Today at 4:07 PM
太誇張了 真的
DataLab.

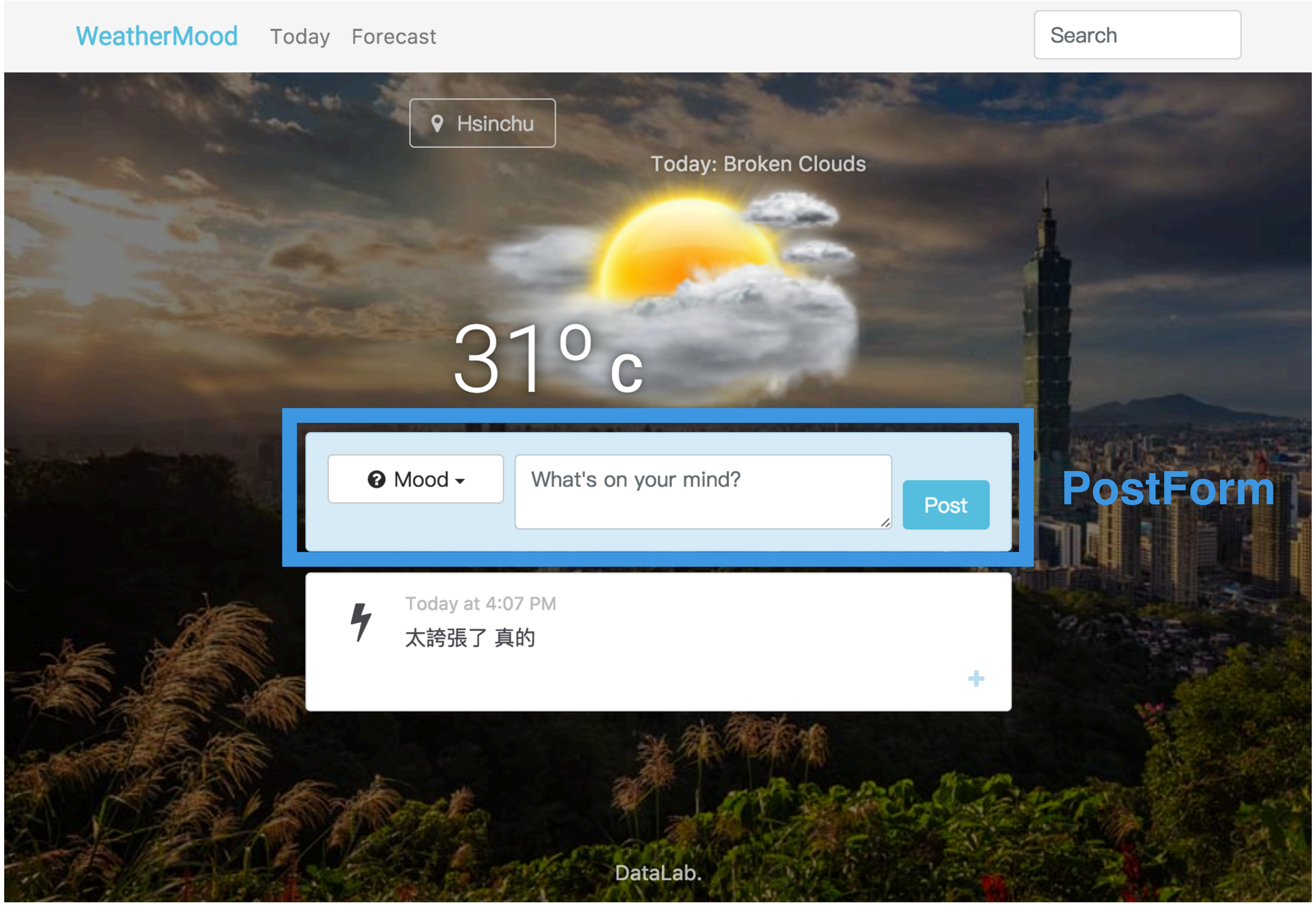

WeatherMood
Today
Forecast
Main
Search
Hsinchu
Today
Today: Broken Clouds
31° c
Mood
What's on your mind?
Post
PostForm
Today at 4:07 PM
太誇張了 真的
DataLab.


WeatherMood
Today
Forecast
Main
Search
Hsinchu
Today: Broken Clouds
31° c
Today
Mood
What's on your mind?
Post
PostForm
Today at 4:07 PM
太誇張了 真的
PostList
DataLab.

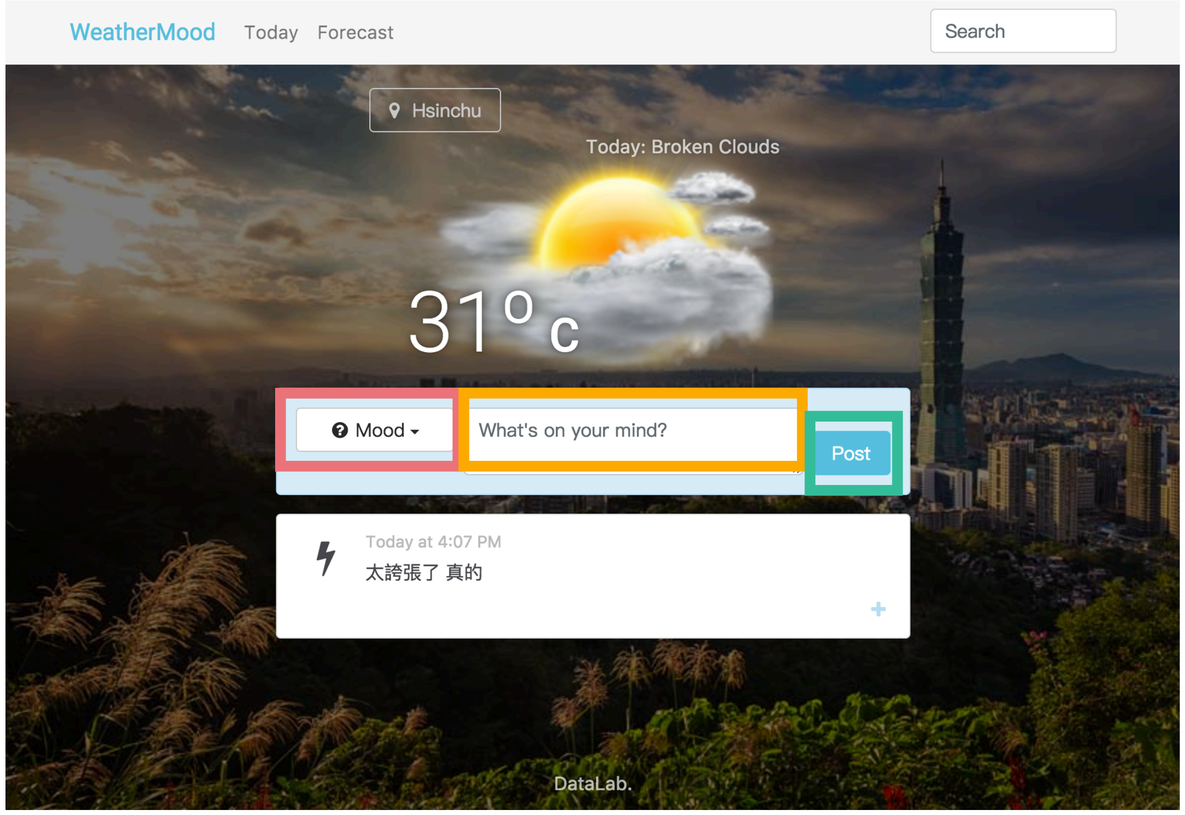

WeatherMood
Today
Forecast
Main
Search
WeatherFrom
Hsinchu
Today
Today: Broken Clouds
31° c
Mood
What's on your mind?
Post
PostForm
Today at 4:07 PM
太誇張了 真的
PostList
DataLab.


WeatherMood
Today
Forecast
Main
Search
WeatherFrom
Hsinchu
Today
Today: Broken Clouds
31° c
Mood
What's on your mind?
Post
PostForm
Today at 4:07 PM
太誇張了 真的
PostItem
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
PostItem
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 ${inputDang
            <div className='mood align-self-start'>
                <ButtonDropdown type='buttom' isOpen={moodToggle} toggle={this.handleMoodToggle}>
                    <DropdownToggle className='mood-toggle' type='button' caret color="secondary">
                        <i className={getMoodIcon(mood)}></i> {
                            mood === 'na' ? 'Mood' : mood
                        }
                    </DropdownToggle>
                    <DropdownMenu>
                        <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Clear')}><i
                        <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Clouds')}><
                        <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Drizzle')}>
                        <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Rain')}><i
                        <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Thunder')}>
                        <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Snow')}><i
                        <DropdownItem type='button' onClick={() => this.handleDropdownSelect('Windy')}><i
                    </DropdownMenu>
                </ButtonDropdown>
            </div>
            <Input className='input' type='textarea' getRef={el => {this.inputEl = el}} value={this.state
            <Button className='btn-post align-self-end' color="info" onClick={this.handlePost}>Post</Butt
        </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='buttom' isOpen={moodToggle
                        <DropdownToggle className='mood-toggle' type
                            <i className={getMoodIcon(mood)}></i>&nb
                                mood === 'na' ? 'Mood' : mood
                            }
```

PostForm - Dropdown

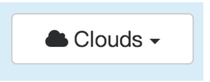
```
handleDropdownSelect(mood) {
    this.setState({mood: mood});
}
```

```
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='buttom' isOpen={moodToggle
                        <DropdownToggle className='mood-toggle' type
                            <i className={getMoodIcon(mood)}></i>&nb
                                mood === 'na' ? '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='buttom' isOpen={moodToggle
                        <DropdownToggle className='mood-toggle' type
                            <i className={getMoodIcon(mood)}></i>&nb
                                mood === 'na' ? 'Mood' : mood
                            }
```



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

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

PostForm - Post

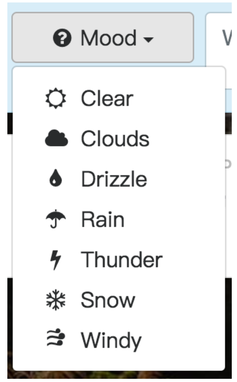
```
handlePost() {
    if (this.state.mood === 'na') {
        this.setState({moodToggle: true});
        return;
    }
    if (!this.state.inputValue) {
        this.setState({inputDanger: true});
        return;
    }

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

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

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'
    });
}
```

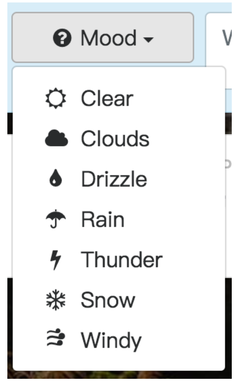
```
<ButtonDropdown type='buttom' isOpen={moodToggle} toggle={this.handleMoodToggle}>
    <DropdownToggle className="mood-toggle" type='button' caret color="secondary">
        <i className={getMoodIcon(mood)}></i> {
            mood === 'na' ? 'Mood' : mood
        }
```



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'
    });
}
```

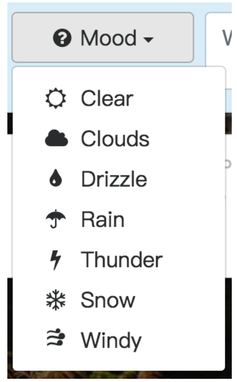
```
<ButtonDropdown type='buttom' isOpen={moodToggle} toggle={this.handleMoodToggle}>
    <DropdownToggle className= mood-toggle type= button' caret color="secondary">
        <i className={getMoodIcon(mood)}></i> {
            mood === 'na' ? 'Mood' : mood
        }
```



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'
    });
}
```

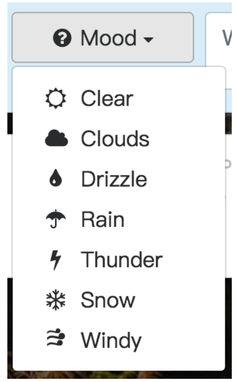
```
<ButtonDropdown type='buttom' isOpen={moodToggle} toggle={this.handleMoodToggle}>
    <DropdownToggle className= mood-toggle type= button' caret color="secondary">
        <i className={getMoodIcon(mood)}></i> {
            mood === 'na' ? 'Mood' : mood
        }
```



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

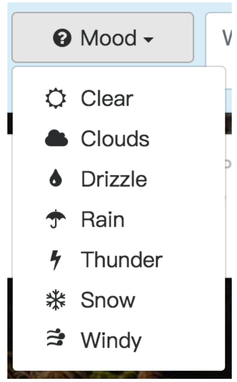



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';

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

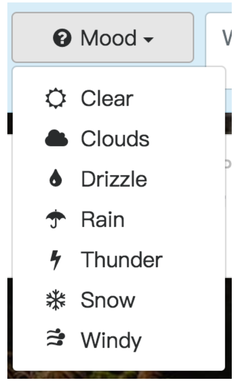



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';

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

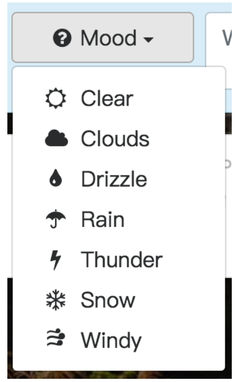



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';

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

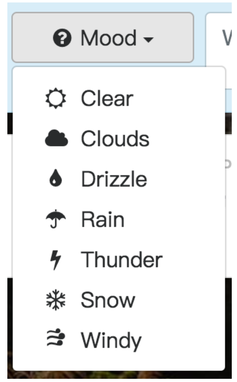



```
const inputDanger = this.state.inputDanger ? 'has-danger' : '';

return (
    <div className='post-form'>
        <Alert color='info' className={`d-flex flex-column flex-sm-row justify-content-center ${inputDanger}`}>
```

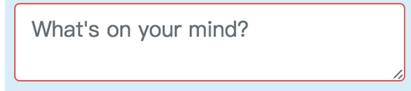
PostForm - Post

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




```
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'
    });
}
```

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);
        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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

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

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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

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

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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

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

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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

```
<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

```
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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

```
<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

```
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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

```
<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

```
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);
        inputValue: '',
        mood: 'na'
    });
}
```

Today.jsx

```
<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

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);
    });
}
```

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);
    });
}
```

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;
};
```

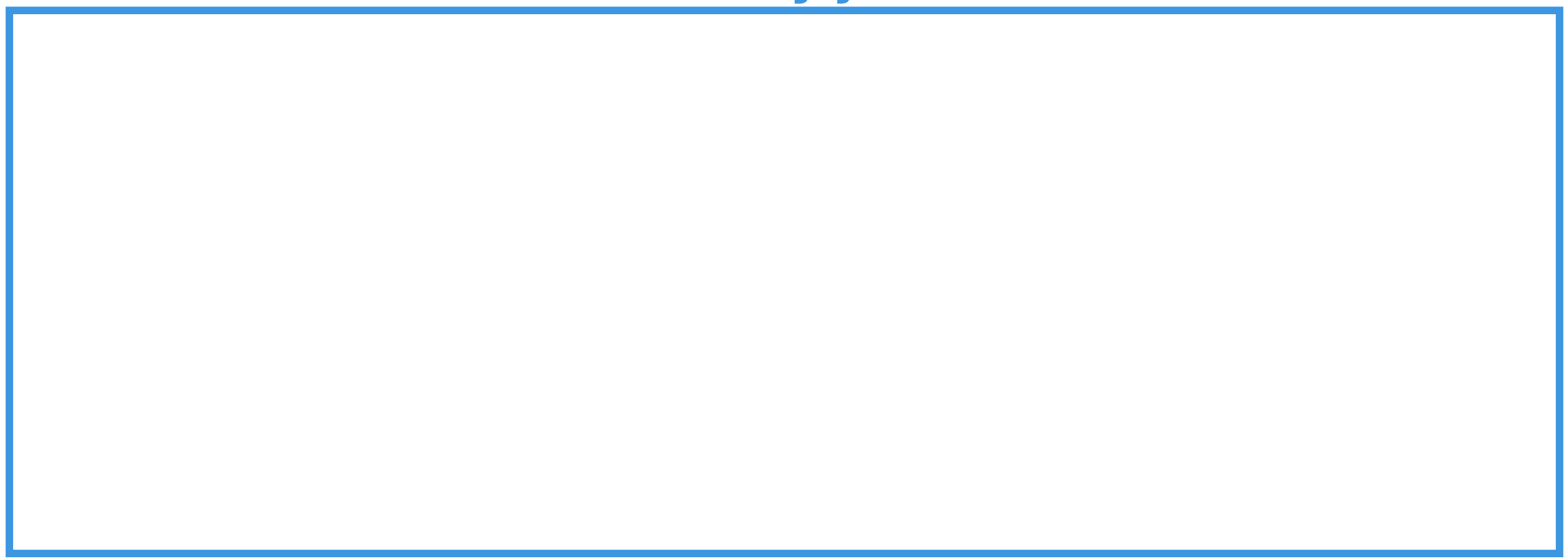
listPosts

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);
    });
}
```

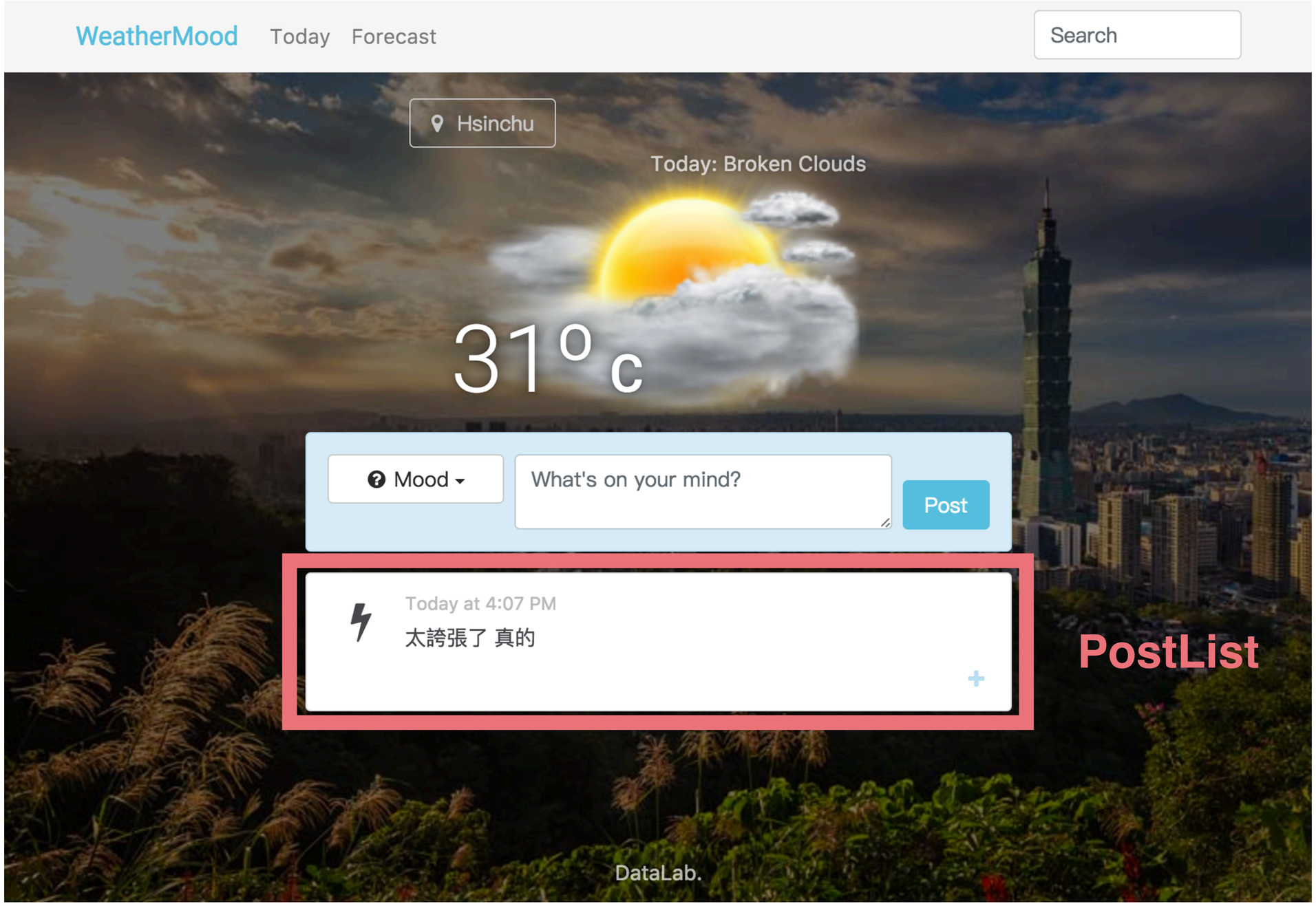
```
// Simulated server-side code
function _listPosts(searchText = '') {
    let postString = localStorage.getItem(postKey);
    let posts = postString ? JSON.parse(postString) : [];
    if (posts.length > 0 && searchText) {
        posts = posts.filter(p => {
            return p.text.toLocaleLowerCase().indexOf(searchText.toLowerCase())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


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
PostList
DataLab.

PostList

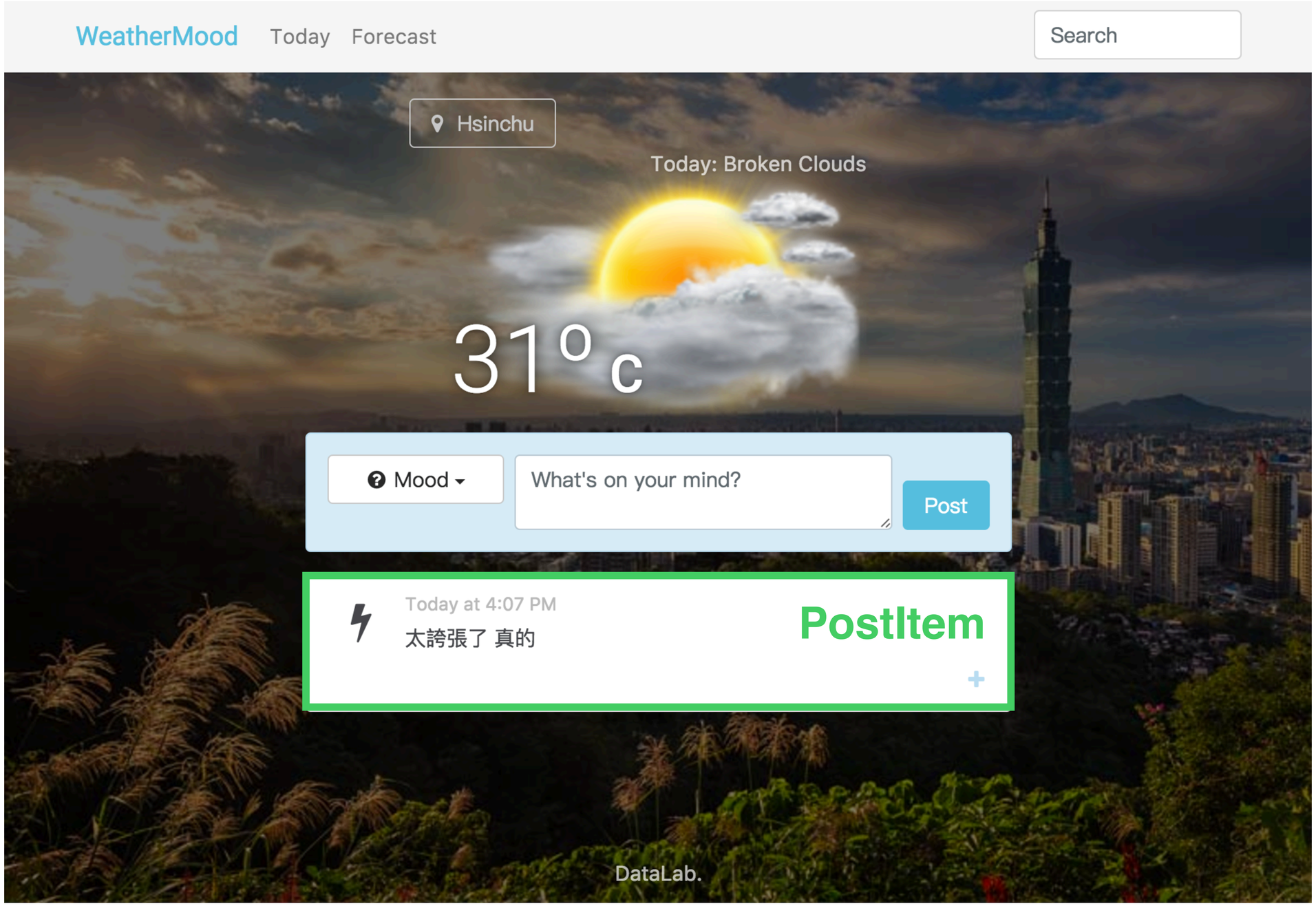
```
render() {
    const {posts} = this.props;

    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>
    );
}
```

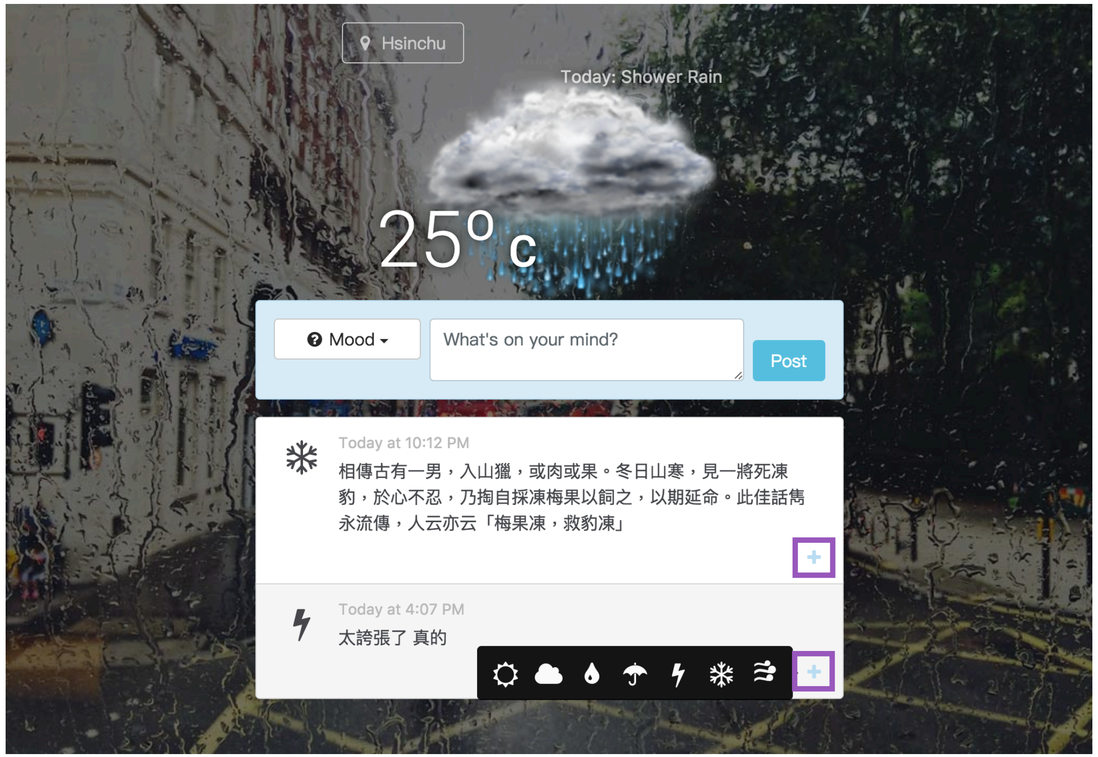

WeatherMood
Today
Forecast
Search
Hsinchu
Today: Broken Clouds
31° c
Mood
What's on your mind?
Post
Today at 4:07 PM
太誇張了 真的
PostItem
DataLab.

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> 
    <i className={`vote-tooltip ${getMoodIcon('Clouds')}`} onClick={() => this.handleVote('Clouds')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Drizzle')}`} onClick={() => this.handleVote('Drizzle')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Rain')}`} onClick={() => this.handleVote('Rain')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Thunder')}`} onClick={() => this.handleVote('Thunder')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Snow')}`} onClick={() => this.handleVote('Snow')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Windy')}`} onClick={() => this.handleVote('Windy')}></i>
</Tooltip>
```

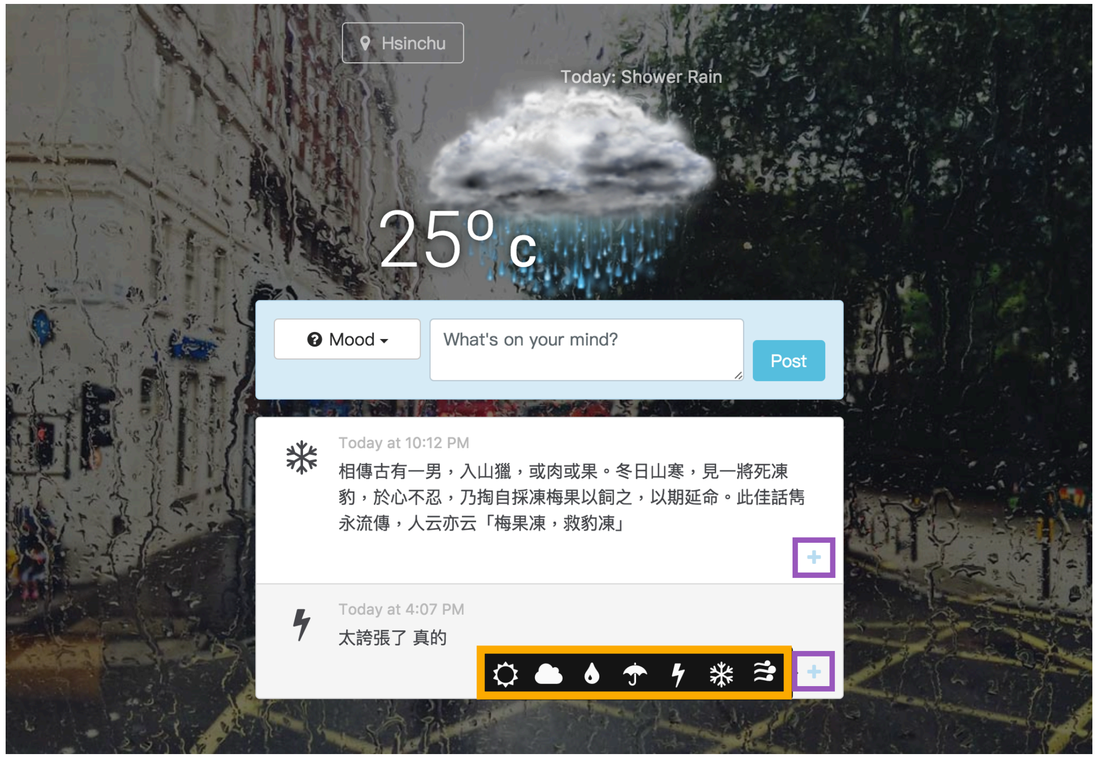
PostItem

```
<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> 
    <i className={`vote-tooltip ${getMoodIcon('Clouds')}`} onClick={() => this.handleVote('Clouds')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Drizzle')}`} onClick={() => this.handleVote('Drizzle')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Rain')}`} onClick={() => this.handleVote('Rain')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Thunder')}`} onClick={() => this.handleVote('Thunder')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Snow')}`} onClick={() => this.handleVote('Snow')}></i> 
    <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> 
    <i className={`vote-tooltip ${getMoodIcon('Clouds')}`} onClick={() => this.handleVote('Clouds')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Drizzle')}`} onClick={() => this.handleVote('Drizzle')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Rain')}`} onClick={() => this.handleVote('Rain')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Thunder')}`} onClick={() => this.handleVote('Thunder')}></i> 
    <i className={`vote-tooltip ${getMoodIcon('Snow')}`} onClick={() => this.handleVote('Snow')}></i> 
    <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

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

render()

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

PostItem

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

render()

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

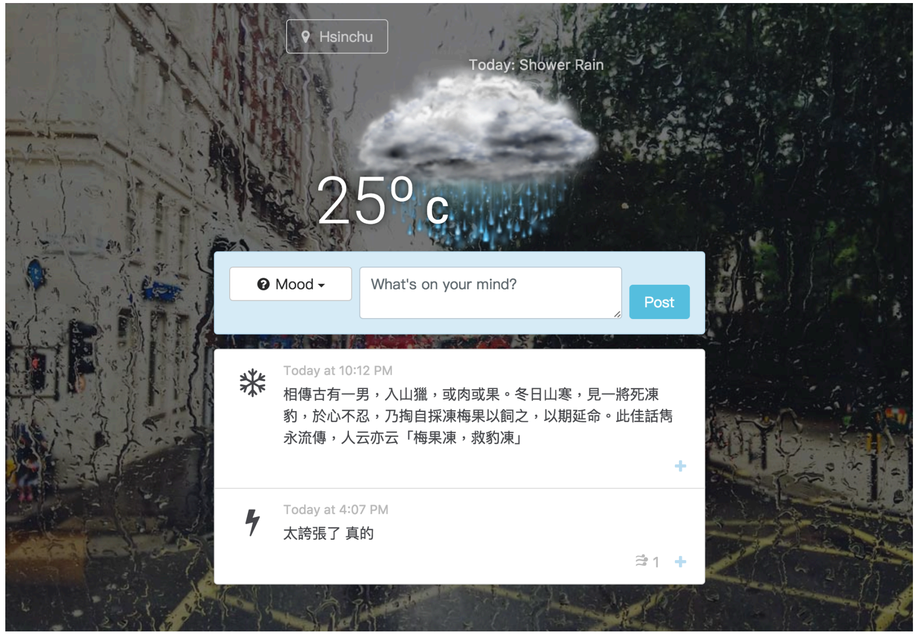
```
<div className='vote d-flex justify-content-end'>
    <div className='vote-results'>
        {clearVotes > 0 && (<span><i className={getMoodIcon('Clear')}></i> {clearVotes}  </span>)}
        {cloudsVotes > 0 && <span><i className={getMoodIcon('Clouds')}></i> {cloudsVotes}  </span>}
        {drizzleVotes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

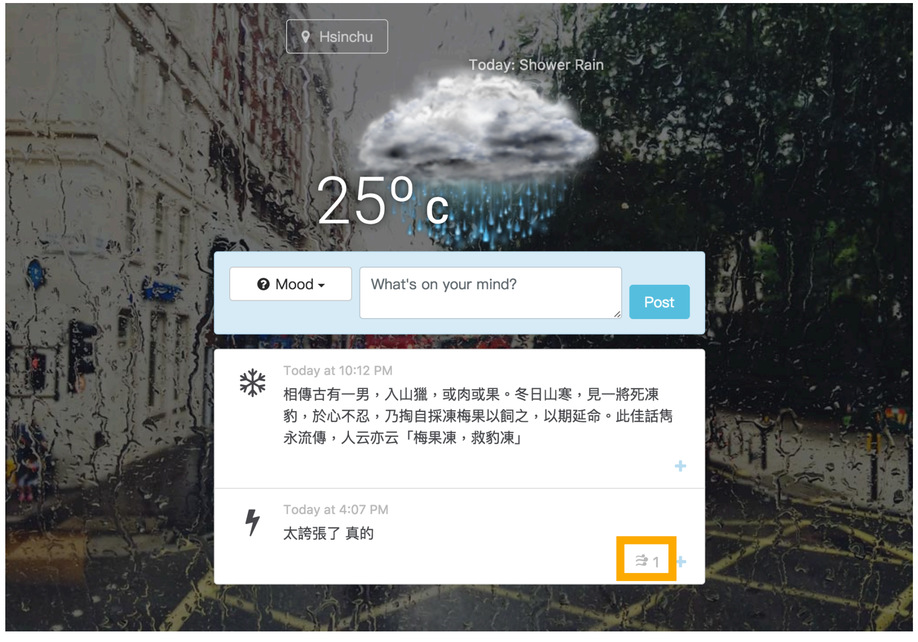


PostItem

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> {clearVotes}  </span>)}
        {cloudsVotes > 0 && <span><i className={getMoodIcon('Clouds')}></i> {cloudsVotes}  </span>}
        {drizzleVotes > 0 && <span><i className={getMoodIcon('Drizzle')}></i> {drizzleVotes}  </span>}
        {rainVotes > 0 && <span><i className={getMoodIcon('Rain')}></i> {rainVotes}  </span>}
        {thunderVotes > 0 && <span><i className={getMoodIcon('Thunder')}></i> {thunderVotes}  </span>}
        {snowVotes > 0 && <span><i className={getMoodIcon('Snow')}></i> {snowVotes}  </span>}
        {windyVotes > 0 && <span><i className={getMoodIcon('Windy')}></i> {windyVotes}  </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

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

PostList.jsx

handleVote

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

PostList.jsx

render()

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

handleVote

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

PostList.jsx

render()

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

handleVote

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

PostList.jsx

render()

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

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

handleVote

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

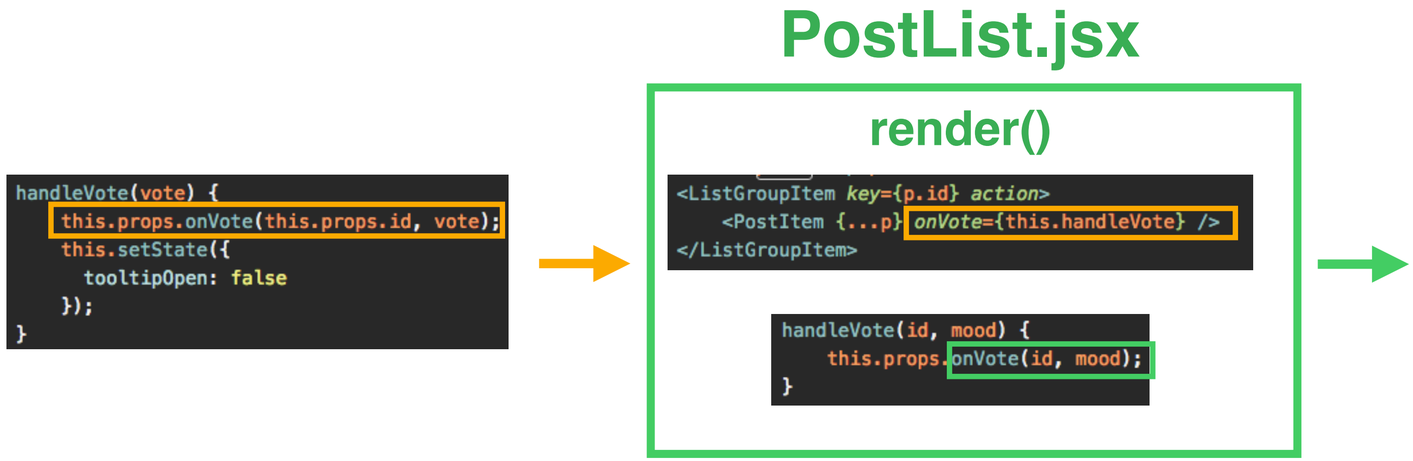
PostList.jsx

render()

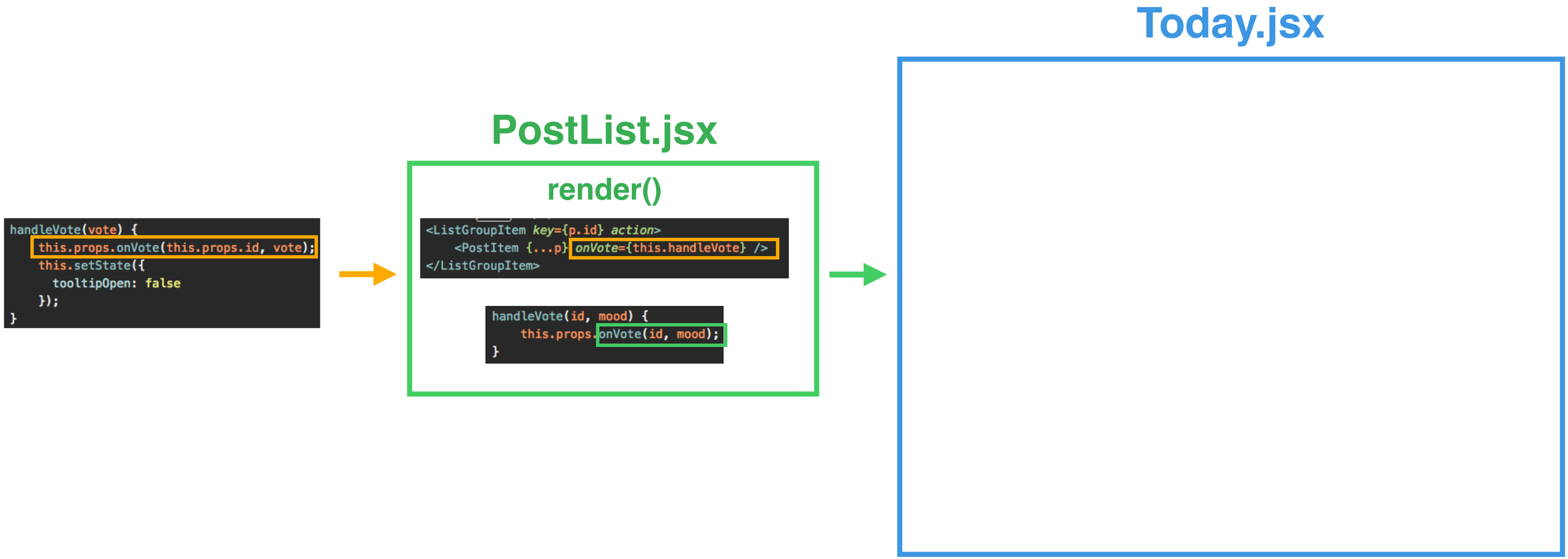
```
<ListGroupItem key={p.id} action>
    <PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>
```

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

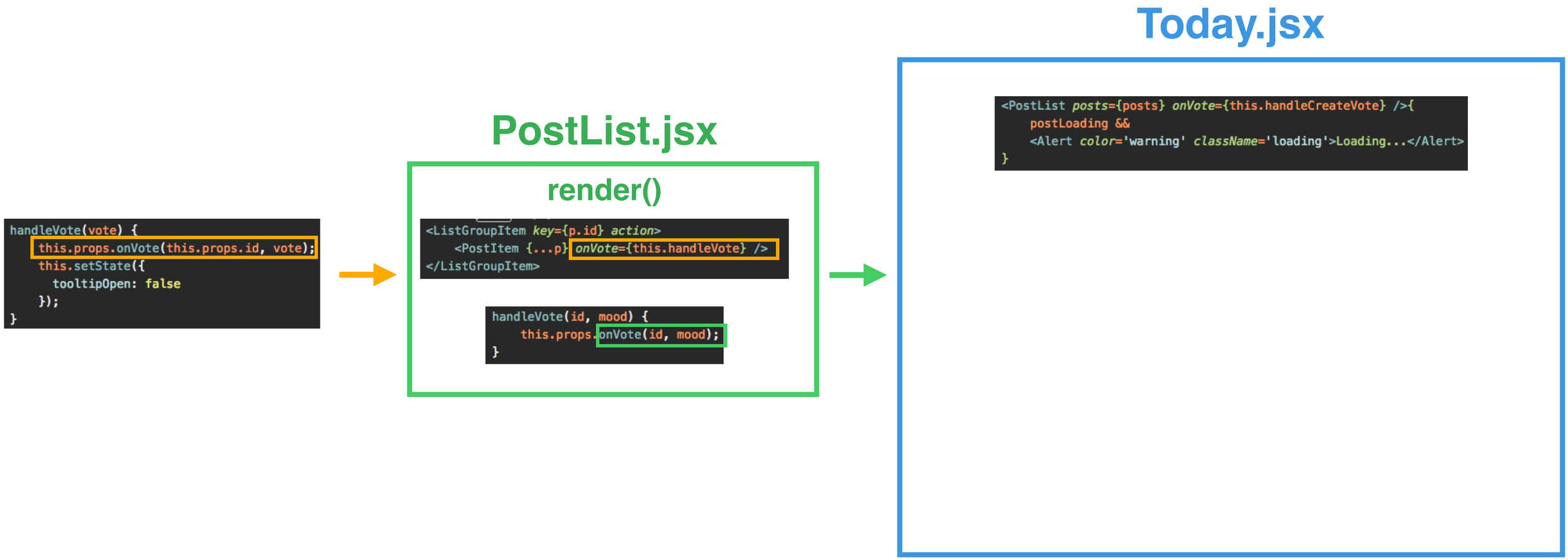
handleVote


handleVote(vote) {
this.props.onVote(this.props.id, vote);
this.setState({
tooltipOpen: false
});
}
PostList.jsx
render()
<ListGroupItem key={p.id} action>
<PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>
handleVote(id, mood) {
this.props.onVote(id, mood);
}

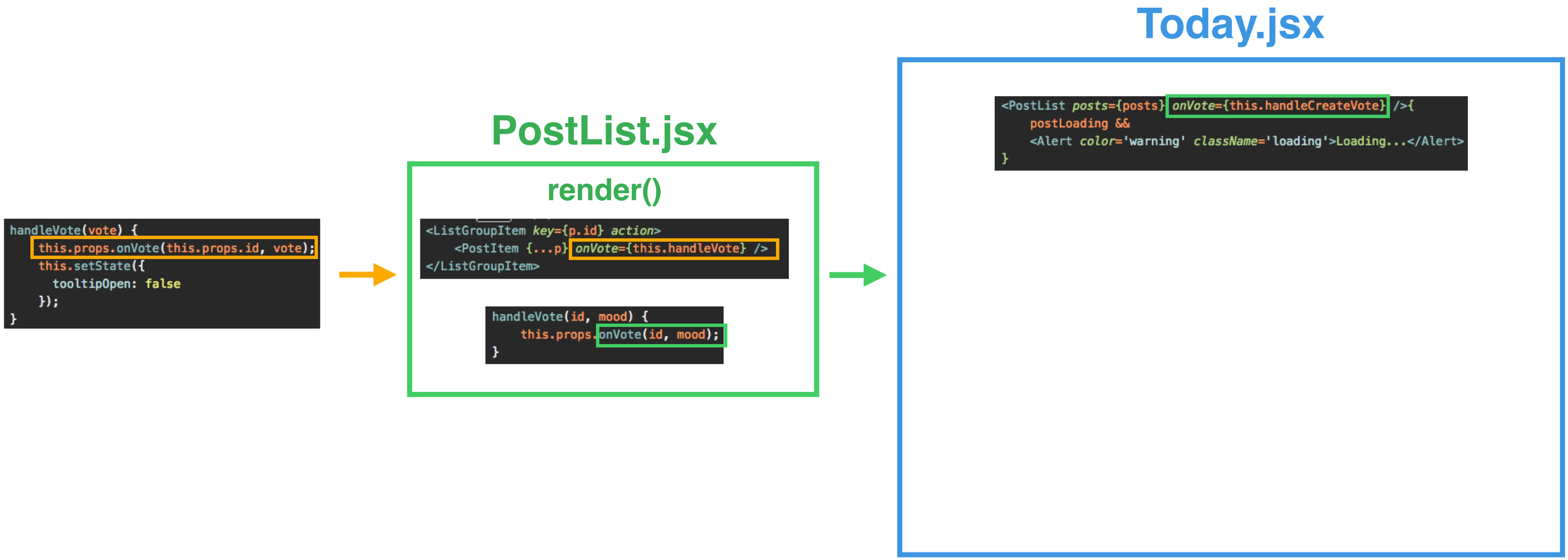
handleVote


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);
}

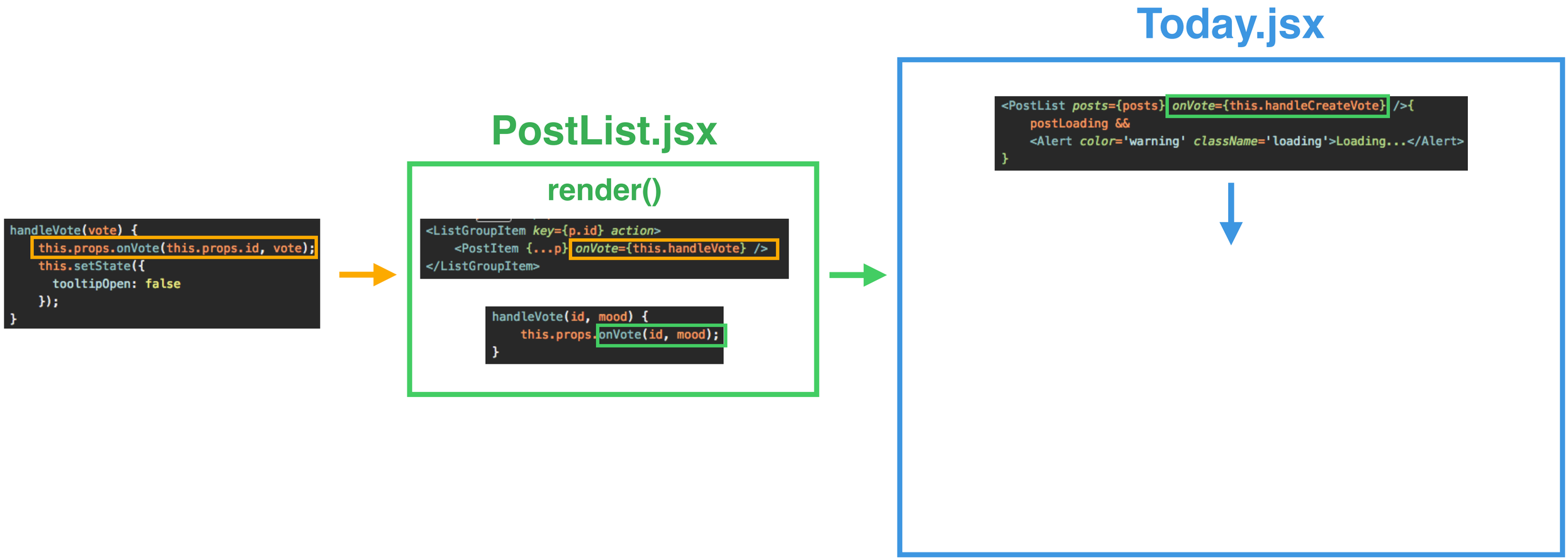
handleVote


Today.jsx
<PostList posts={posts} onVote={this.handleCreateVote} />{
postLoading &&
<Alert color='warning' className='loading'>Loading...</Alert>
}
PostList.jsx
render()
<ListGroupItem key={p.id} action>
<PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>
handleVote(id, mood) {
this.props.onVote(id, mood);
}
handleVote(vote) {
this.props.onVote(this.props.id, vote);
this.setState({
tooltipOpen: false
});
}

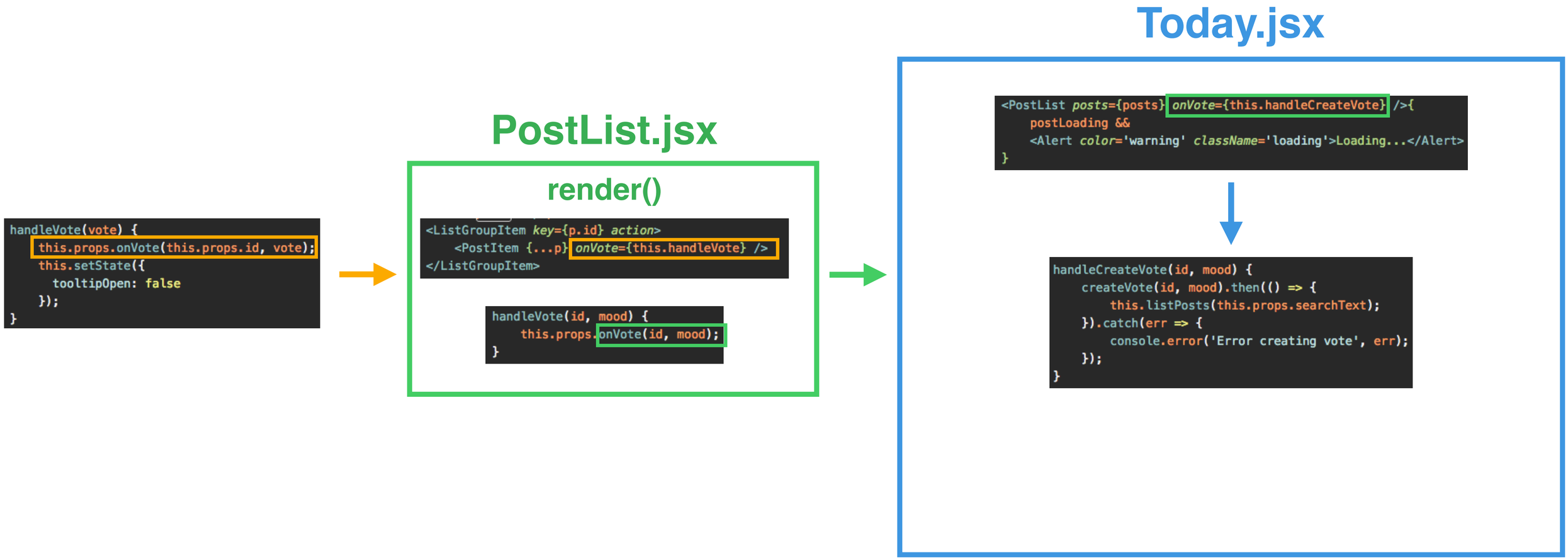
handleVote


Today.jsx
<PostList posts={posts} onVote={this.handleCreateVote} />{
postLoading &&
<Alert color='warning' className='loading'>Loading...</Alert>
}
PostList.jsx
render()
<ListGroupItem key={p.id} action>
<PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>
handleVote(id, mood) {
this.props.onVote(id, mood);
}
handleVote(vote) {
this.props.onVote(this.props.id, vote);
this.setState({
tooltipOpen: false
});
}

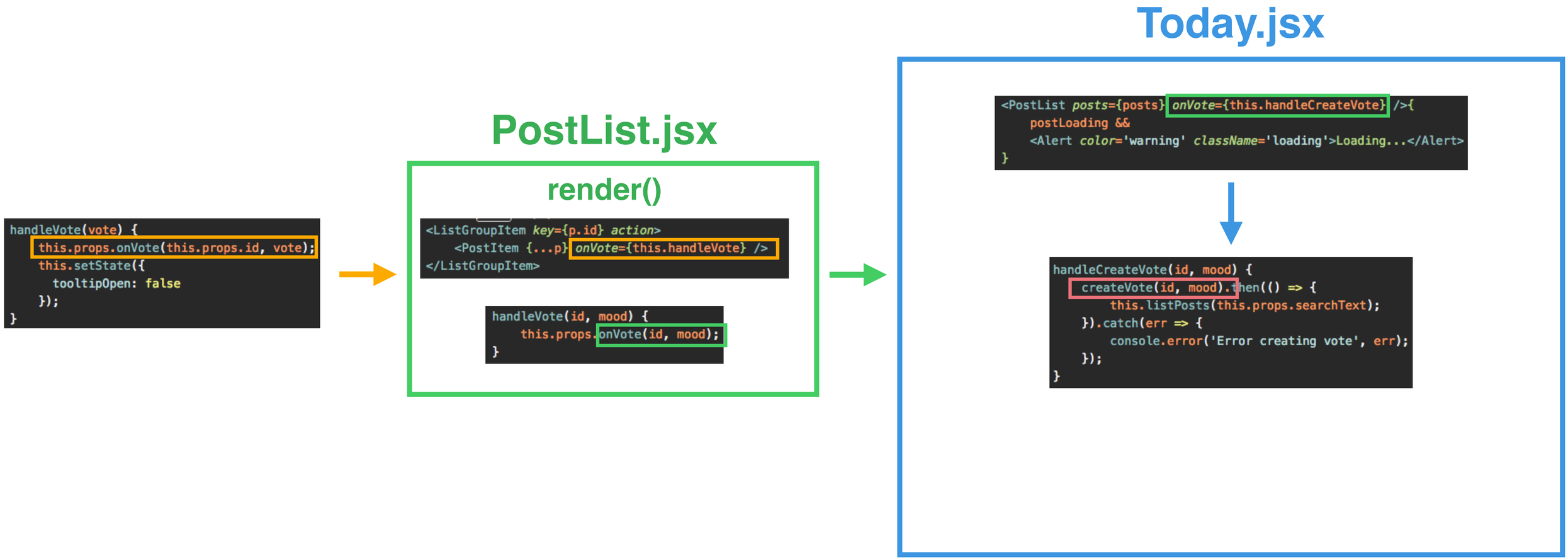
handleVote


Today.jsx
<PostList posts={posts} onVote={this.handleCreateVote} />{
postLoading &&
<Alert color='warning' className='loading'>Loading...</Alert>
}
PostList.jsx
render()
<ListGroupItem key={p.id} action>
<PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>
handleVote(id, mood) {
this.props.onVote(id, mood);
}
handleVote(vote) {
this.props.onVote(this.props.id, vote);
this.setState({
tooltipOpen: false
});
}

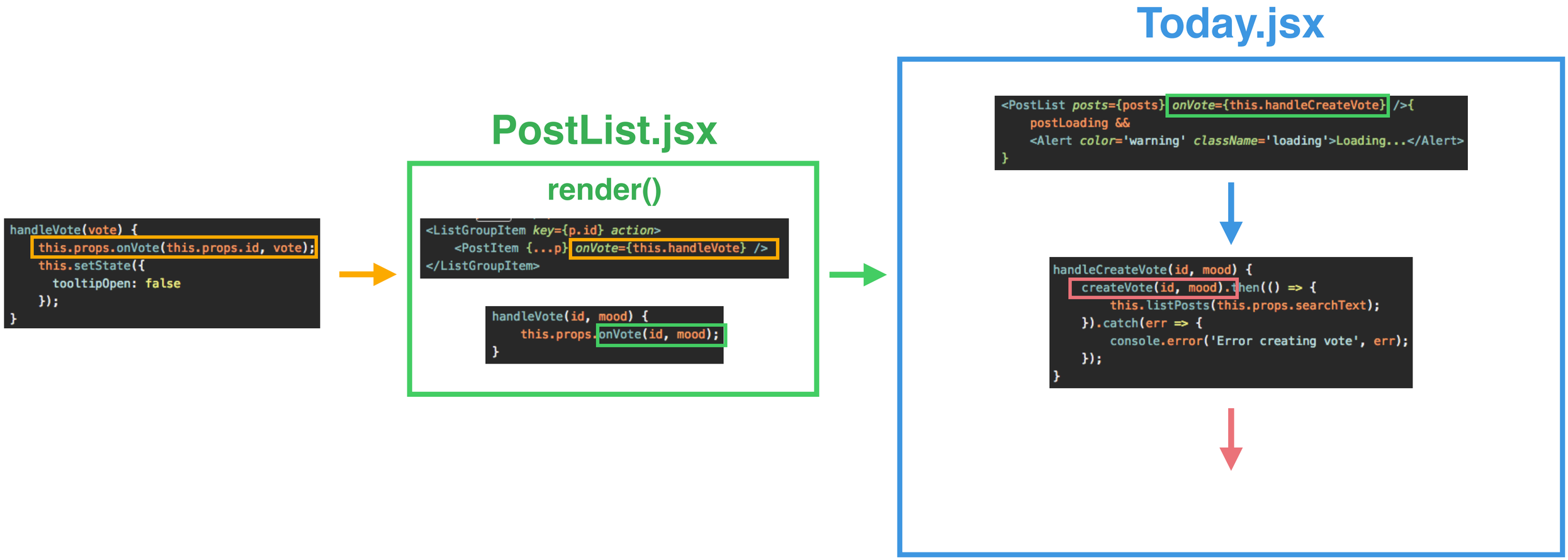
handleVote


Today.jsx
<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);
});
}
PostList.jsx
render()
<ListGroupItem key={p.id} action>
<PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>
handleVote(id, mood) {
this.props.onVote(id, mood);
}
handleVote(vote) {
this.props.onVote(this.props.id, vote);
this.setState({
tooltipOpen: false
});
}

handleVote


Today.jsx
<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);
});
}
PostList.jsx
render()
<ListGroupItem key={p.id} action>
<PostItem {...p} onVote={this.handleVote} />
</ListGroupItem>
handleVote(id, mood) {
this.props.onVote(id, mood);
}
handleVote(vote) {
this.props.onVote(this.props.id, vote);
this.setState({
tooltipOpen: false
});
}

handleVote


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

PostList.jsx

render()

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

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

Today.jsx

```
<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