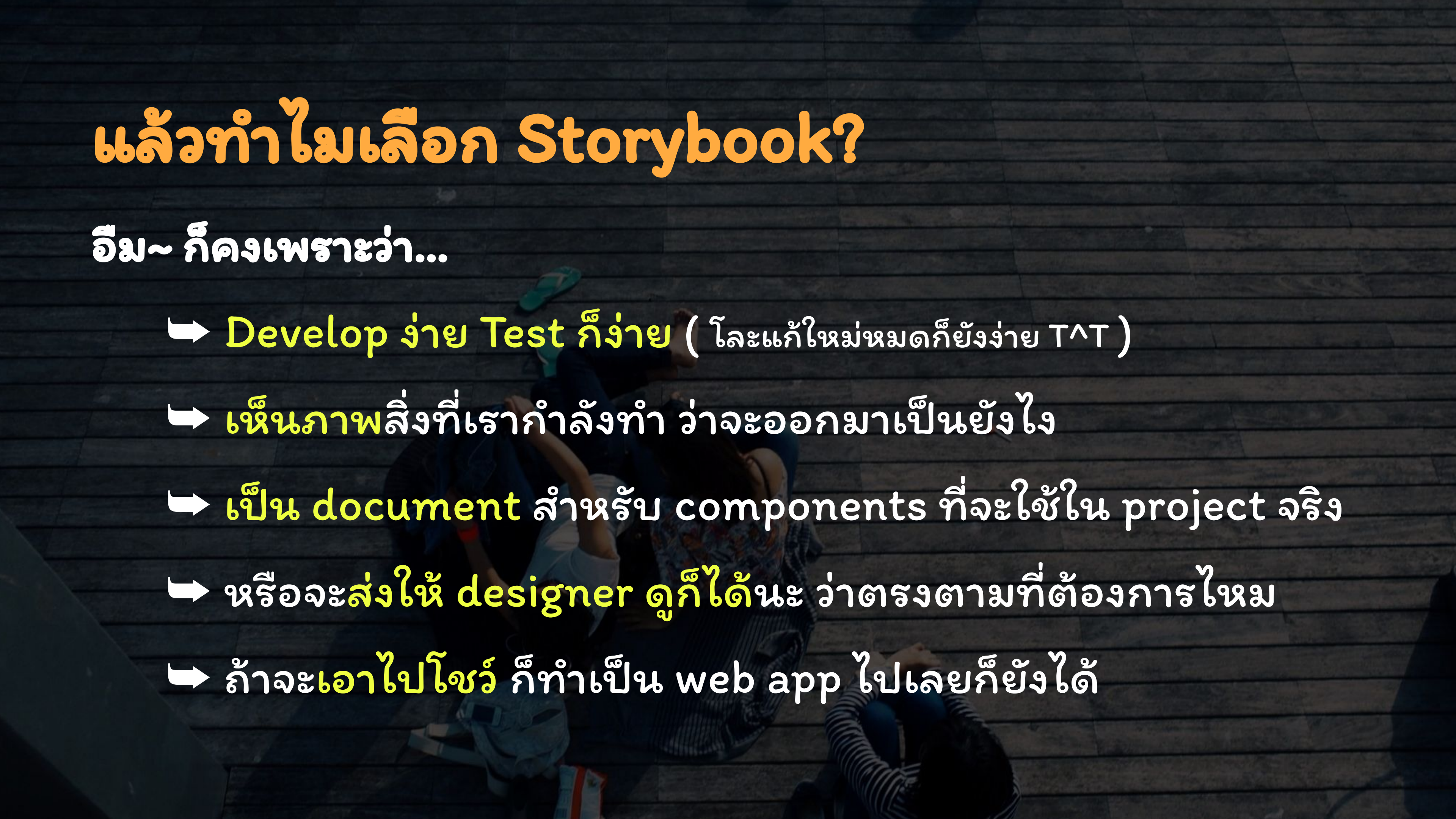
- Storybook

Storybook คือ...?

“Storybook is a user interface development environment and playground for UI components.” - by storybook.js.org

→ Storybook คือ tool ที่ช่วยให้เขียนและ test UI component ต่างๆของ framework เช่น Vue หรือ React ที่ base on ความเป็น component ได้ง่ายและเร็วขึ้น!

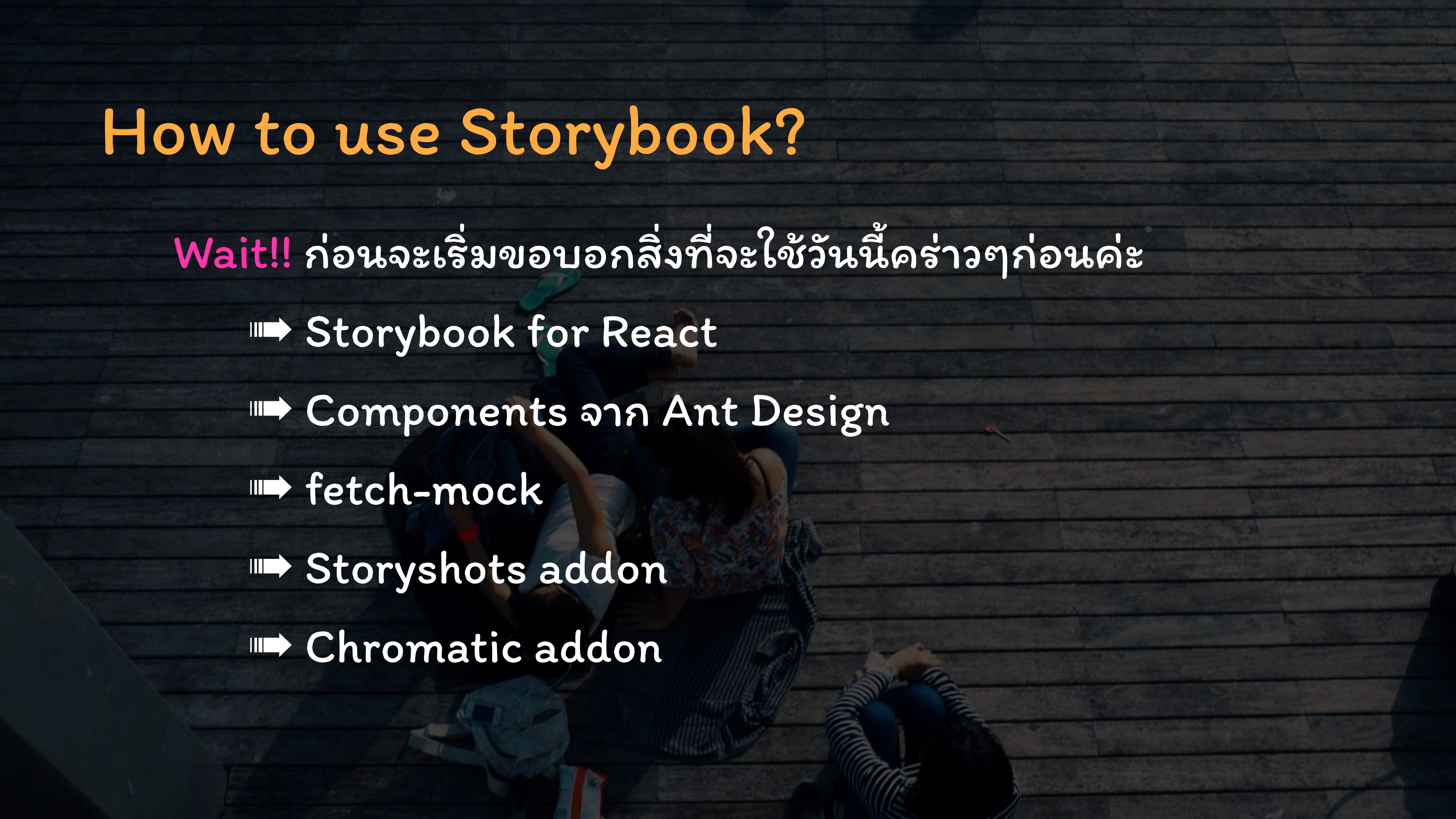
→ Storybook เหมือน ‘สมุดภาพ’ ของ developer ที่จะทำ UI



แล้วทำไมเลือก Storybook?

อืม~ ก็คงเพราะว่า...

- ➥ Develop ง่าย Test ก็ง่าย (โละแก้ใหม่หมดก็ยังง่าย T^T)
- ➥ เห็นภาพสิ่งที่เรากำลังทำ ว่าจะออกมาเป็นยังไง
- ➥ เป็น document สำหรับ components ที่จะใช้ใน project จริง
- ➥ หรือจะส่งให้ designer ดูก็ได้นะ ว่าตรงตามที่ต้องการไหม
- ➥ ถ้าจะเอาไปโชว์ ก็ทำเป็น web app ไปเลยก็ยังได้



How to use Storybook?

Wait!! ก่อนจะเริ่มขอบอกสิ่งที่จะใช้วันนี้คร่าวๆก่อนค่ะ

- ➠ Storybook for React
- ➠ Components จาก Ant Design
- ➠ fetch-mock
- ➠ Storyshots addon
- ➠ Chromatic addon

เริ่มจาก...

1. Create your application:

```
$ npx create-react-app storyb --typescript
$ cd storyb
```

2. Add Storybook:

```
$ npx -p @storybook/cli sb init
```

→ At this step, your project should already have...
‘.storybook’ folder in root directory and ‘stories’ folder in src folder.

3. Quickly check that the various environments of our application are working properly:

```
# Start the component explorer on port 9009:
$ yarn run storybook

# Run the frontend app proper on port 3000:
$ yarn start

# Run the test runner (Jest) in a terminal:
$ yarn test
```

Hmm, any problem in previous step?

```
$ yarn run storybook
```

Possible error/warning: Error: ENOENT: no such file or directory, stat 'path'

Fix by: `$ yarn install` and run `$ yarn run storybook` again

```
$ yarn test
```

and/or

```
$ yarn start
```

Possible error/warning: error Command failed with exit code 1.

Fix by: Create `.env` file in your project and add `SKIP_PREFLIGHT_CHECK=true`

or install watchman in your machine by `$ brew install watchman`

4. Install Ant Design and some devDependencies:

```
# Install Ant Design:
$ yarn add antd
```

Note: You can also use `$ npm install antd`

```
# Install babel-loader, @babel/core and @babel/preset-env:
$ yarn add --dev babel-loader @babel/core @babel/preset-env
```

Note: You can also use

```
$ npm install babel-loader @babel/core @babel/preset-env --save-dev
```

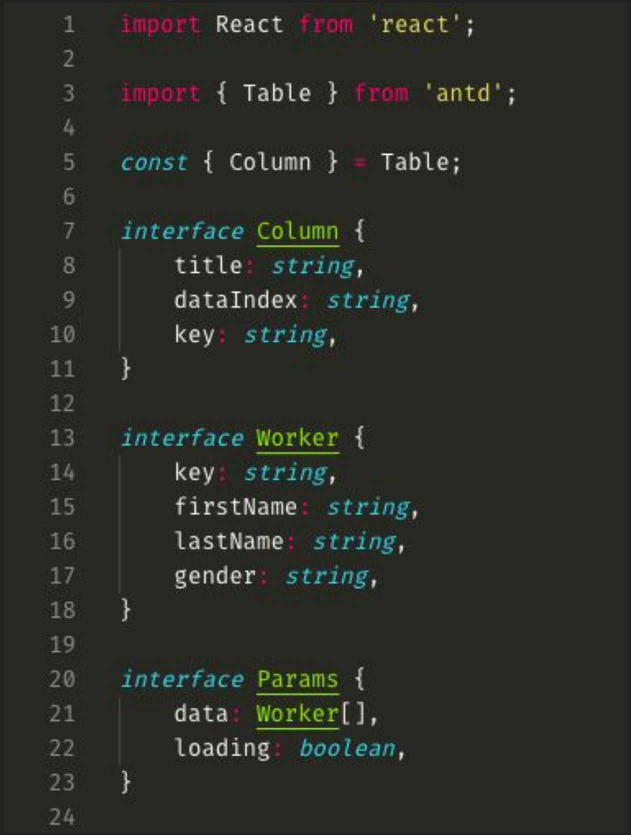
5. Create .babelrc file in your project then…

# Add a few text below into the file

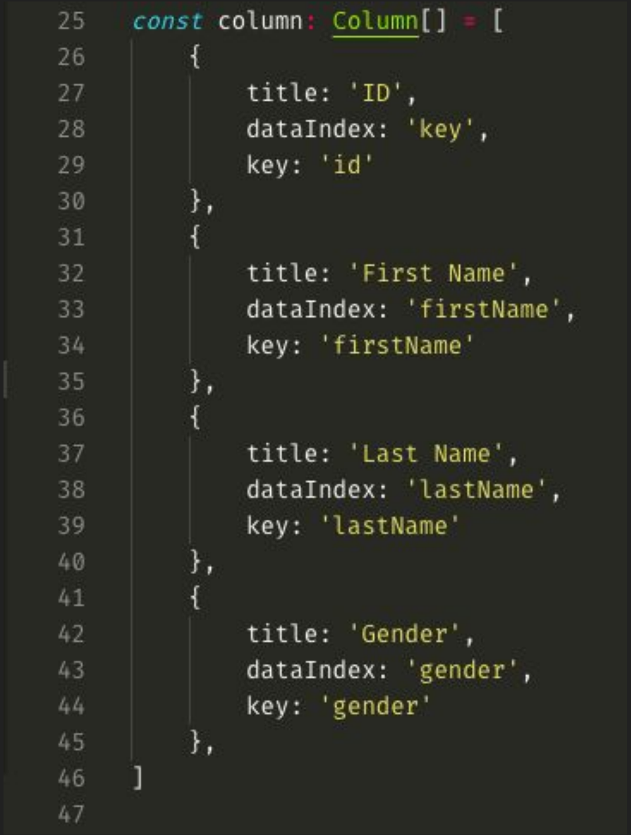
```
{
        “presets” : [ “@babel/preset-env” ]
}
```

Note: Since I installed babel-cli and babel-preset-env and add { “presets”: [“env”] } in .babelrc file but it’s not working in my machine then I use @babel/preset-env instead.

6. Create DataTable.tsx file in ‘src’ folder:



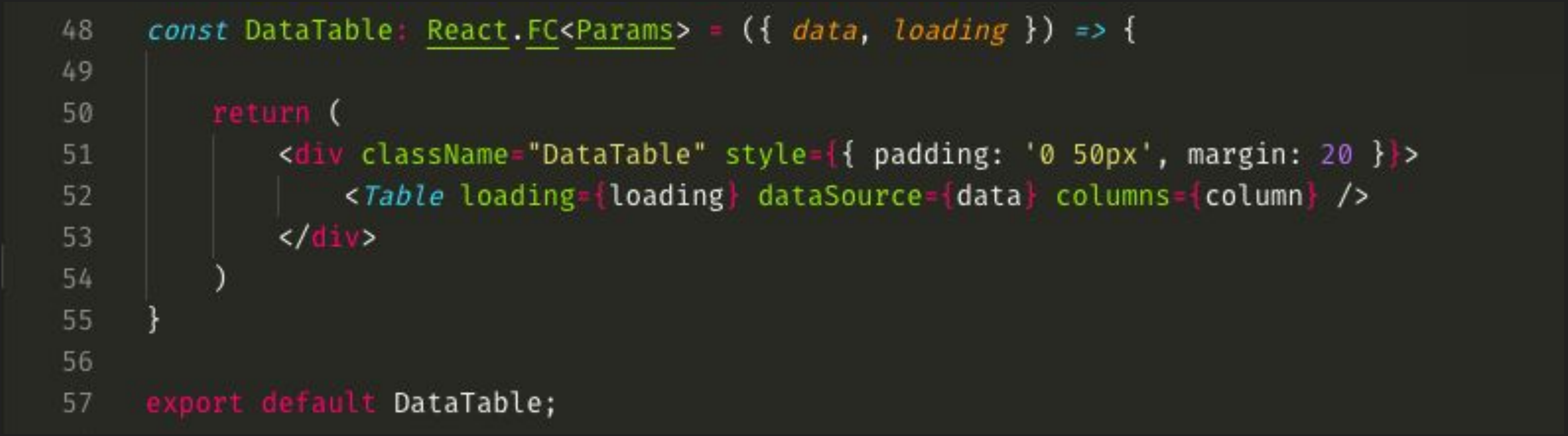
```tsx
import React from 'react';

import { Table } from 'antd';

const { Column } = Table;

interface Column {
    title: string,
    dataIndex: string,
    key: string,
}

interface Worker {
    key: string,
    firstName: string,
    lastName: string,
    gender: string,
}

interface Params {
    data: Worker[],
    loading: boolean,
}

```



```tsx
const column: Column[] = [
    {
        title: 'ID',
        dataIndex: 'key',
        key: 'id'
    },
    {
        title: 'First Name',
        dataIndex: 'firstName',
        key: 'firstName'
    },
    {
        title: 'Last Name',
        dataIndex: 'lastName',
        key: 'lastName'
    },
    {
        title: 'Gender',
        dataIndex: 'gender',
        key: 'gender'
    },
]

```

6. Create DataTable.tsx file in ‘src’ folder:

# Add code below into your file



```
const DataTable: React.FC<Params> = ({ data, loading }) => {

    return (
        <div className="DataTable" style={{ padding: '0 50px', margin: 20 }}>
            <Table loading={loading} dataSource={data} columns={column} />
        </div>
    )
}

export default DataTable;
```

7. Create story for your component in Storybook:

3 ways to do this:

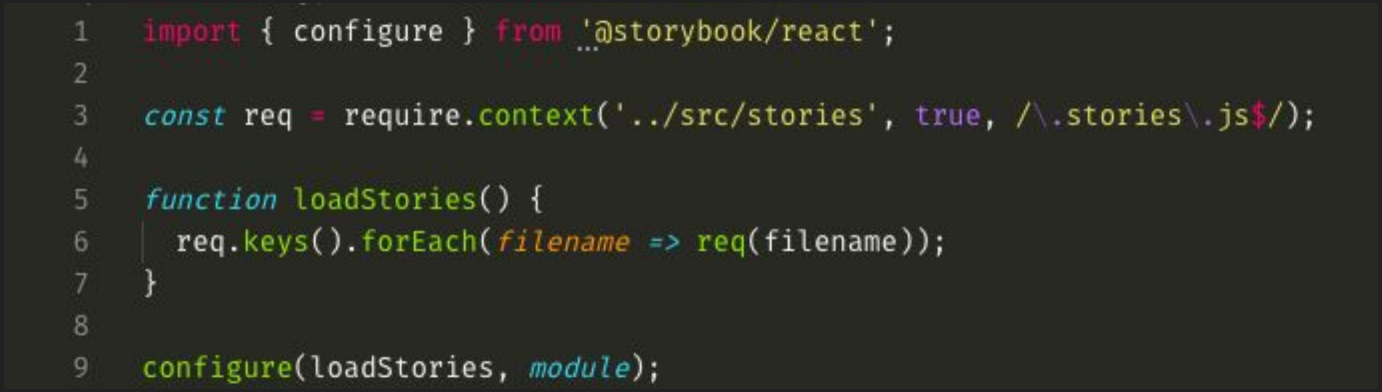
Way 1: Add in index.js file in stories folder that we already have.

Way 2: Create new .js file for your component.

- Create DataTable.js in src/stories/ folder
- Go to .storybook/config.js
- Delete ‘ `require('../src/stories');` ’ in loadStories()
- Add `const req = require.context('../src/stories', true, /.js$/);` above loadStories()
- Add `req.keys().forEach(filename => req(filename));` in loadStories()

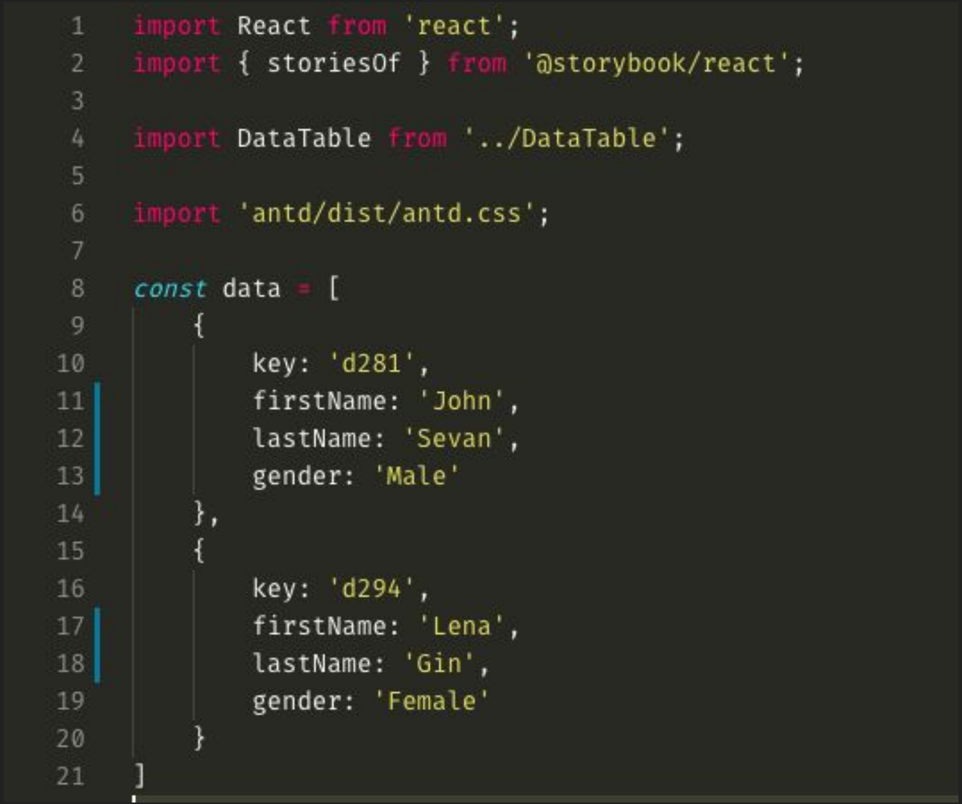
Way 3: Create .stories.js file for each component.

# Similar to Way 2 but we will create DataTable.stories.js and add
const req = require.context('../src/stories', true, /\.stories\.js$/);
above loadStories() in .storybook/config.js instead.
# I'll change index.js ⟶ index.stories.js also.



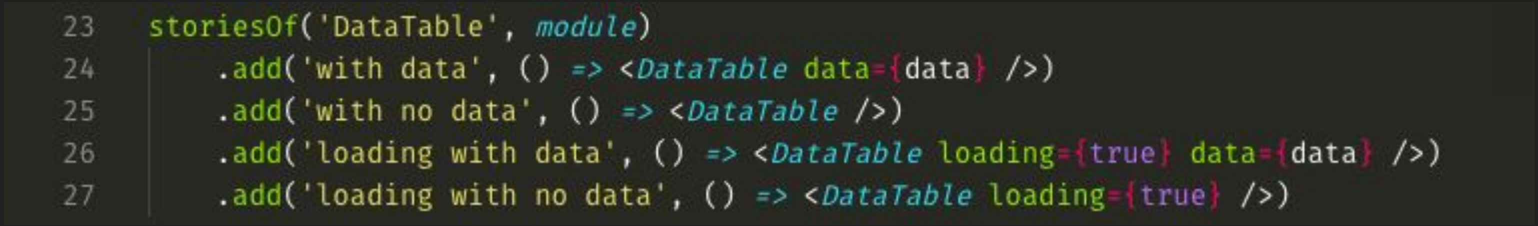
```
import { configure } from '@storybook/react';

const req = require.context('../src/stories', true, /\.stories\.js$/);

function loadStories() {
  req.keys().forEach(filename => req(filename));
}

configure(loadStories, module);
```

8. Add story in DataTable.stories.js file in 'stories' folder:



```
import React from 'react';
import { storiesOf } from '@storybook/react';

import DataTable from '../DataTable';

import 'antd/dist/antd.css';

const data = [
    {
        key: 'd281',
        firstName: 'John',
        lastName: 'Sevan',
        gender: 'Male'
    },
    {
        key: 'd294',
        firstName: 'Lena',
        lastName: 'Gin',
        gender: 'Female'
    }
]
```

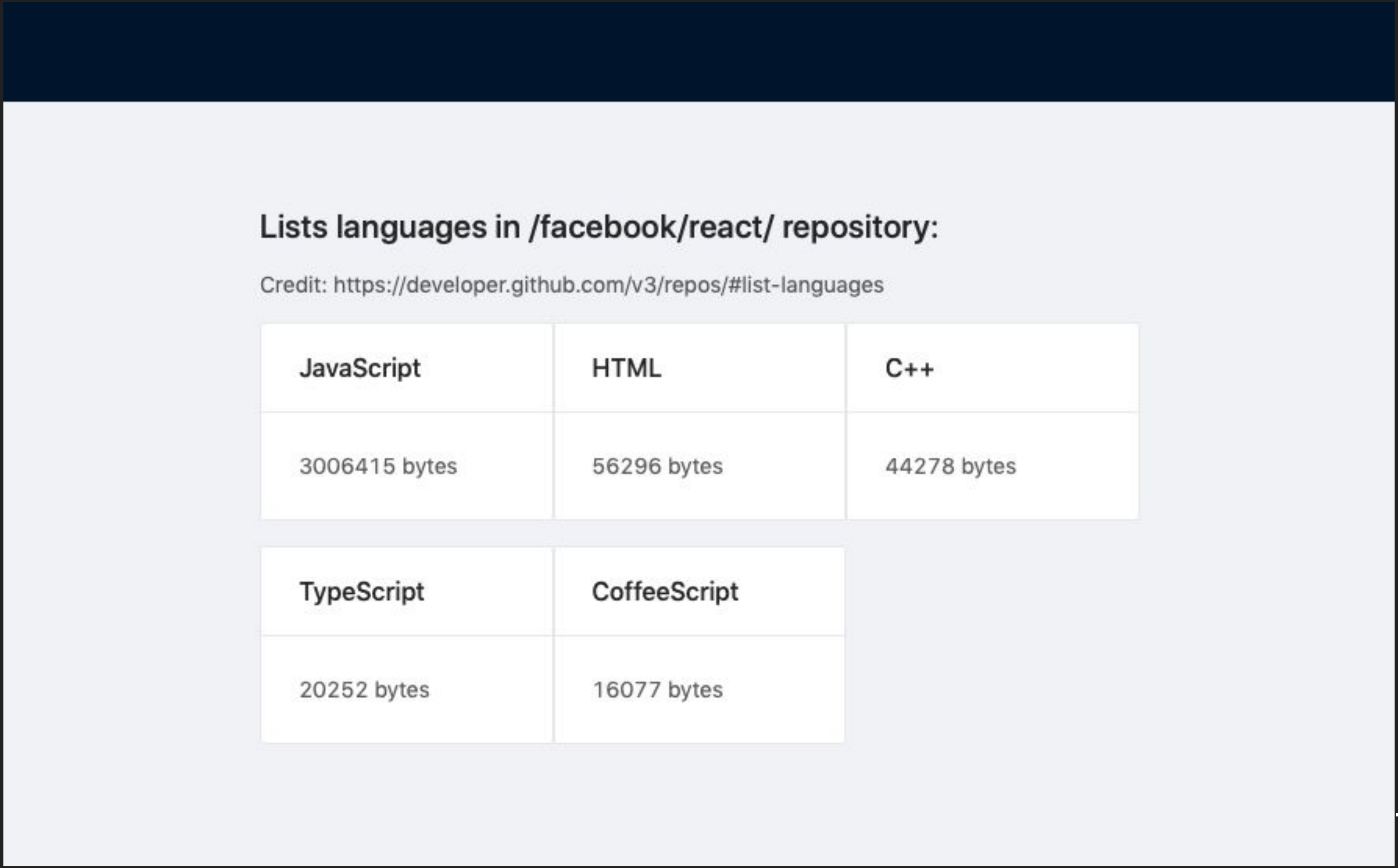
8. Add story in DataTable.stories.js file in 'stories' folder:



```
storiesOf('DataTable', module)
    .add('with data', () => <DataTable data={data} />)
    .add('with no data', () => <DataTable />)
    .add('loading with data', () => <DataTable loading={true} data={data} />)
    .add('loading with no data', () => <DataTable loading={true} />)
```

After added code above, run `$ yarn run storybook`

Next example we'll use fetch-mock to get data from https://api.github.com/repos/facebook/react/languages



Lists languages in /facebook/react/ repository:

Credit: https://developer.github.com/v3/repos/#list-languages

JavaScript	HTML	C++
3006415 bytes	56296 bytes	44278 bytes

TypeScript	CoffeeScript
20252 bytes	16077 bytes

9. Install fetch-mock and other devDependencies:

```
# Install fetch-mock:
$ yarn add --dev fetch-mock
```

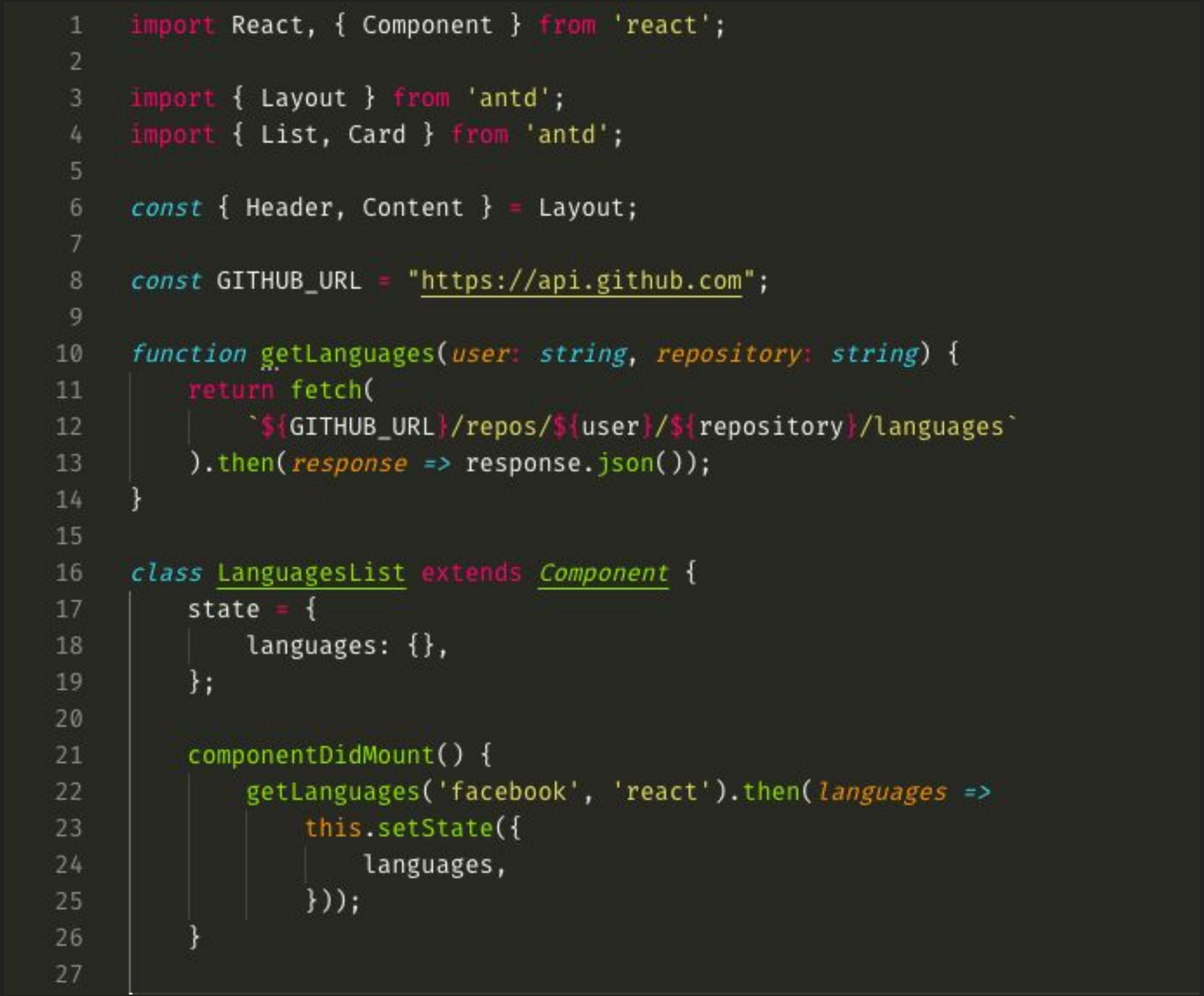
Note: You can also use `$ npm install fetch-mock --save-dev`

```
# Install core-js@2.6.9:
$ yarn add --dev core-js@2.6.9
```

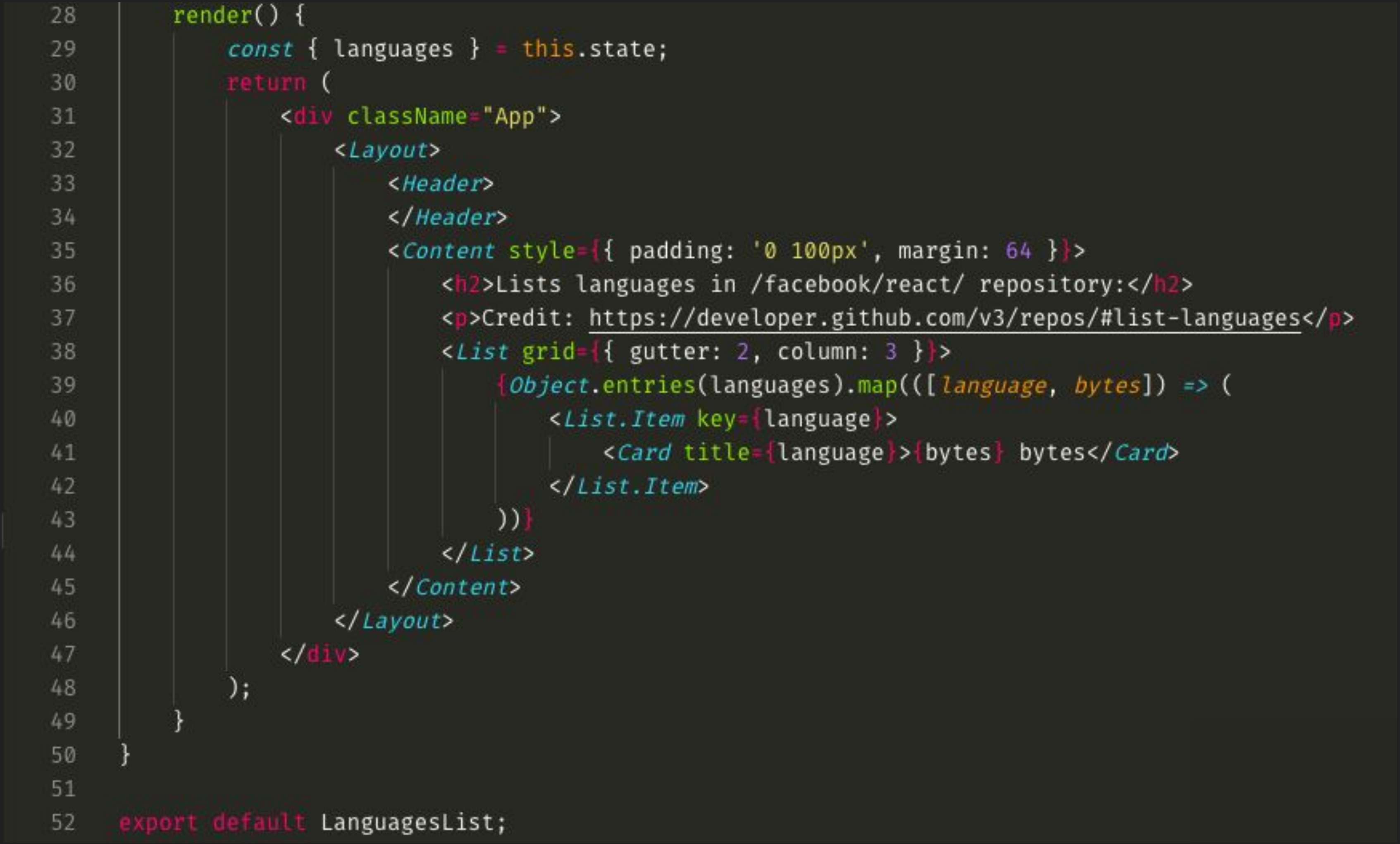
Note: You can also use `$ npm install core-js@2.6.9 --save-dev`

!! WARNING !!: Must be core-js v2.x because fetch-mock doesn’t support core-js v3.x just yet. (Based on an issue in fetch-mock GitHub repos.)

10. Create LanguagesList.tsx file in 'src' folder:

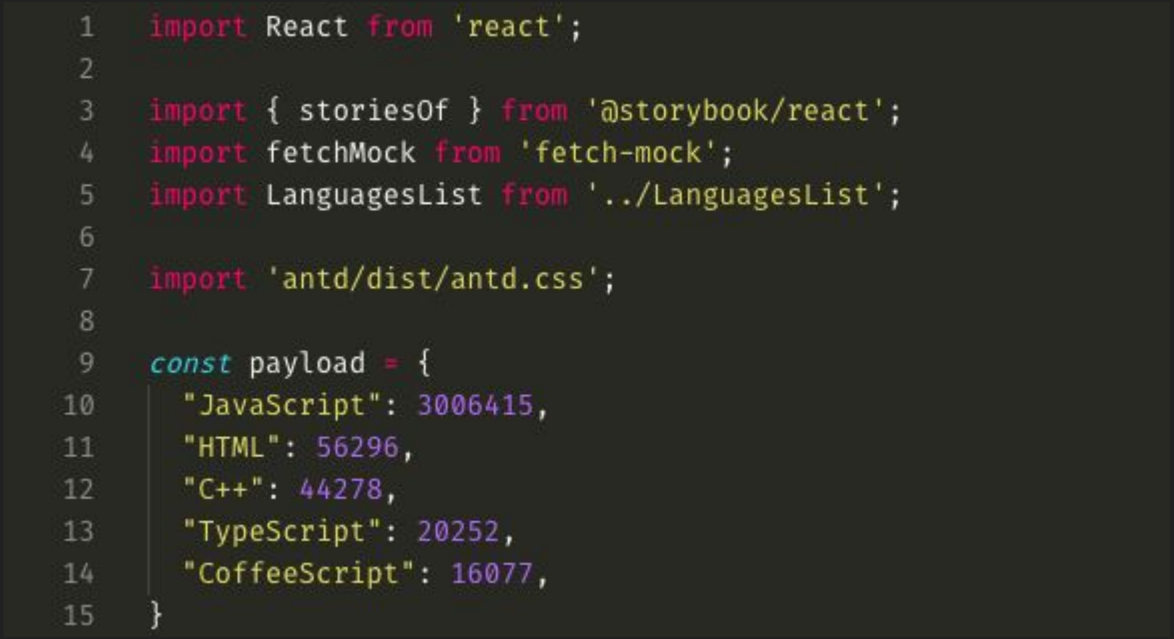


```
import React, { Component } from 'react';

import { Layout } from 'antd';
import { List, Card } from 'antd';

const { Header, Content } = Layout;

const GITHUB_URL = "https://api.github.com";

function getLanguages(user: string, repository: string) {
    return fetch(
        `${GITHUB_URL}/repos/${user}/${repository}/languages`
    ).then(response => response.json());
}

class LanguagesList extends Component {
    state = {
        languages: {},
    };

    componentDidMount() {
        getLanguages('facebook', 'react').then(languages =>
            this.setState({
                languages,
            }));
    }

```

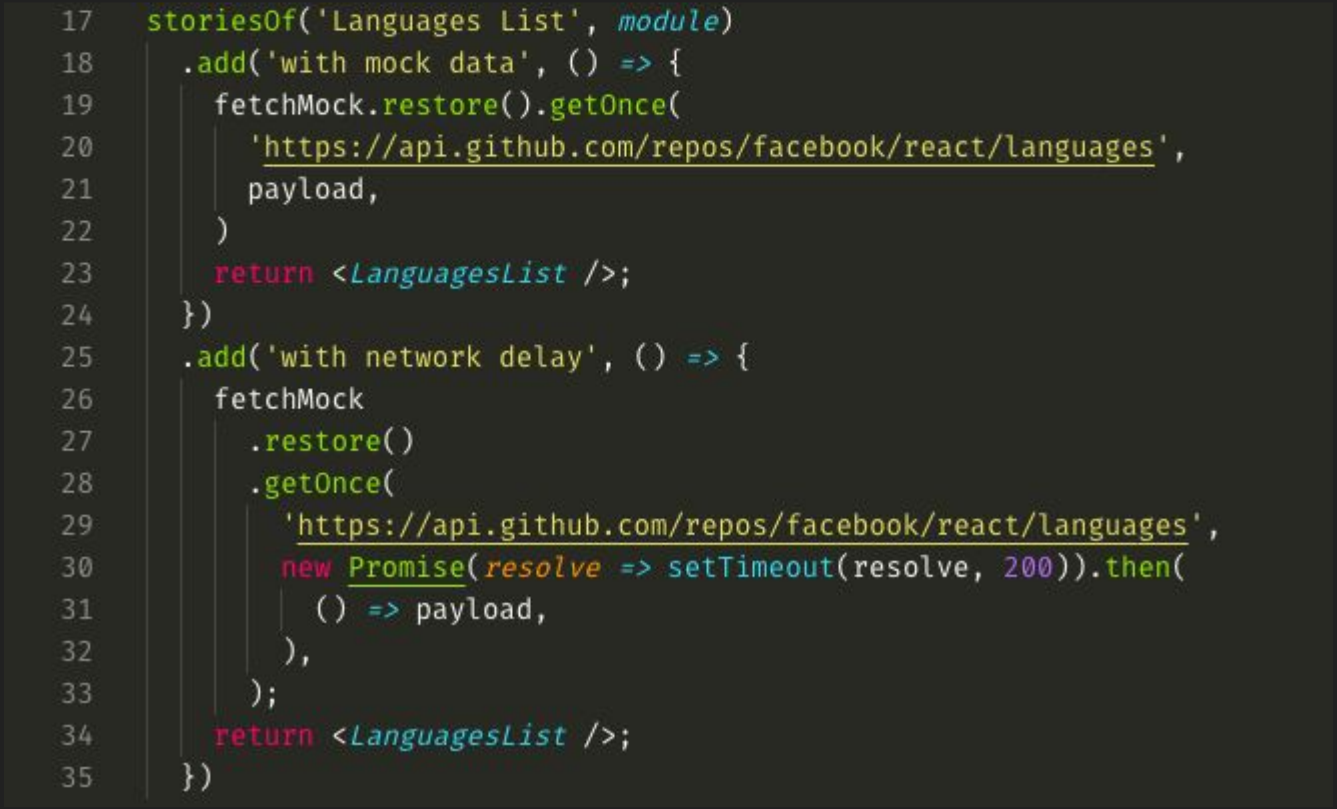


```
    render() {
        const { languages } = this.state;
        return (
            <div className="App">
                <Layout>
                    <Header>
                    </Header>
                    <Content style={{ padding: '0 100px', margin: 64 }}>
                        <h2>Lists languages in /facebook/react/ repository:</h2>
                        <p>Credit: https://developer.github.com/v3/repos/#list-languages</p>
                        <List grid={{ gutter: 2, column: 3 }}>
                            {Object.entries(languages).map(([language, bytes]) => (
                                <List.Item key={language}>
                                    <Card title={language}>{bytes} bytes</Card>
                                </List.Item>
                            ))}
                        </List>
                    </Content>
                </Layout>
            </div>
        );
    }
}

export default LanguagesList;
```

11. Create LanguagesList.stories.js file in 'stories' folder:



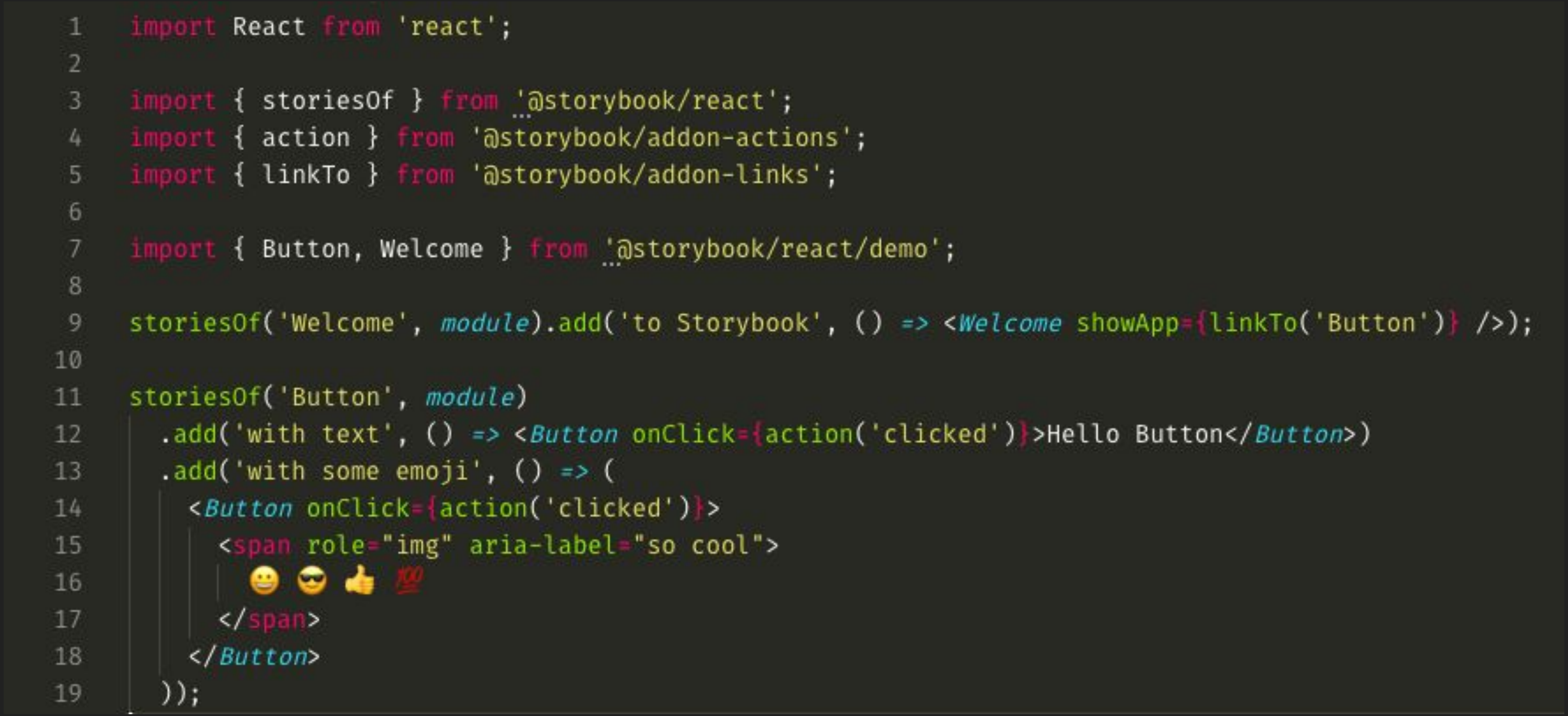
```javascript
import React from 'react';

import { storiesOf } from '@storybook/react';
import fetchMock from 'fetch-mock';
import LanguagesList from '../LanguagesList';

import 'antd/dist/antd.css';

const payload = {
  "JavaScript": 3006415,
  "HTML": 56296,
  "C++": 44278,
  "TypeScript": 20252,
  "CoffeeScript": 16077,
}
```



```
storiesOf('Languages List', module)
  .add('with mock data', () => {
    fetchMock.restore().getOnce(
      'https://api.github.com/repos/facebook/react/languages',
      payload,
    )
    return <LanguagesList />;
  })
  .add('with network delay', () => {
    fetchMock
      .restore()
      .getOnce(
        'https://api.github.com/repos/facebook/react/languages',
        new Promise(resolve => setTimeout(resolve, 200)).then(
          () => payload,
        ),
      );
    return <LanguagesList />;
  })
```

After added code above, run $ yarn run storybook

More addons in src/stories/index.stories.js:



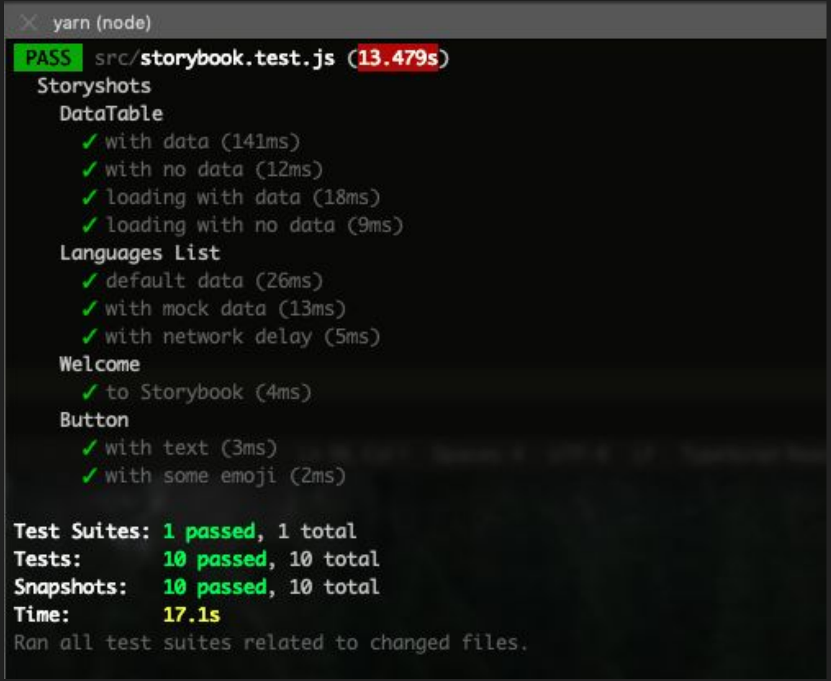
```jsx
import React from 'react';

import { storiesOf } from '@storybook/react';
import { action } from '@storybook/addon-actions';
import { linkTo } from '@storybook/addon-links';

import { Button, Welcome } from '@storybook/react/demo';

storiesOf('Welcome', module).add('to Storybook', () => <Welcome showApp={linkTo('Button')} />);

storiesOf('Button', module)
  .add('with text', () => <Button onClick={action('clicked')}>Hello Button</Button>)
  .add('with some emoji', () => (
    <Button onClick={action('clicked')}>
      <span role="img" aria-label="so cool">
        😀 😎 👍 💯
      </span>
    </Button>
  ));
```

Testing our Storybook:

3 types of tests:

⊕ Type 1: Snapshot tests

Snapshots tests with Storyshots capture a component's rendered markup. They help us stay abreast of markup changes that cause rendering errors and warnings.



```
yarn (node)
PASS  src/storybook.test.js (13.479s)
  Storyshots
    DataTable
      ✓ with data (141ms)
      ✓ with no data (12ms)
      ✓ loading with data (18ms)
      ✓ loading with no data (9ms)
    Languages List
      ✓ default data (26ms)
      ✓ with mock data (13ms)
      ✓ with network delay (5ms)
    Welcome
      ✓ to Storybook (4ms)
    Button
      ✓ with text (3ms)
      ✓ with some emoji (2ms)

Test Suites: 1 passed, 1 total
Tests:       10 passed, 10 total
Snapshots:   10 passed, 10 total
Time:        17.1s
Ran all test suites related to changed files.
```

1. Install Storyshots addon and other devDependencies:

```
# Install @storybook/addon-storyshots and other dependencies:
$ yarn add --dev @storybook/addon-storyshots
react-test-renderer require-context.macro
babel-plugin-macros
```

Note: You can also use `$ npm install @storybook/addon-storyshots react-test-renderer require-context.macro babel-plugin-macros --save-dev`

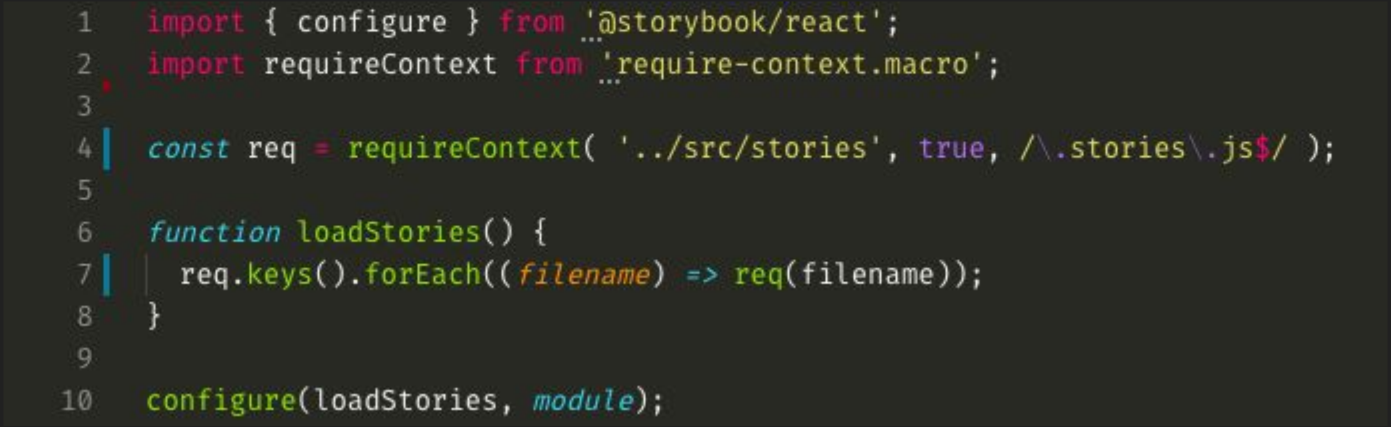
Note-2:

→ Install babel-plugin-macros to ensure that require.context runs in Jest.

→ Install require-context.macro to ensure that we have babel-plugin-macros installed in our project.

2. Edit config.js file in '.storybook' folder:

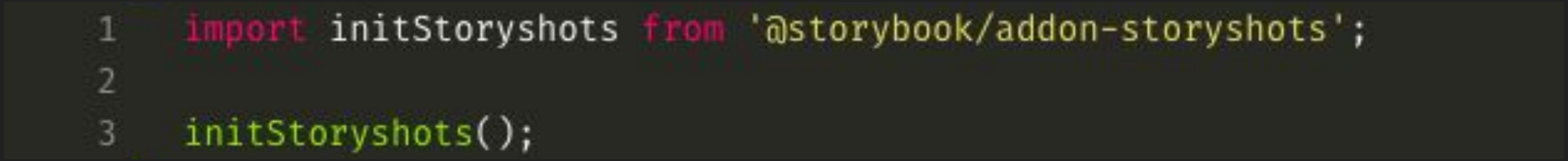
# Go to .storybook/config.js
# Add line 2
# Change line 4 form require.context() ⟶ requireContext()



```
 1  import { configure } from '@storybook/react';
 2  import requireContext from 'require-context.macro';
 3
 4  const req = requireContext( '../src/stories', true, /\.stories\.js$/ );
 5
 6  function loadStories() {
 7    req.keys().forEach((filename) => req(filename));
 8  }
 9
10  configure(loadStories, module);
```

3. Create storybook.test.js file in 'src' folder:

# Add the code below in storybook.test.js file



```
import initStoryshots from '@storybook/addon-storyshots';

initStoryshots();
```

4. Run Snapshots tests:

```
$ yarn test
```

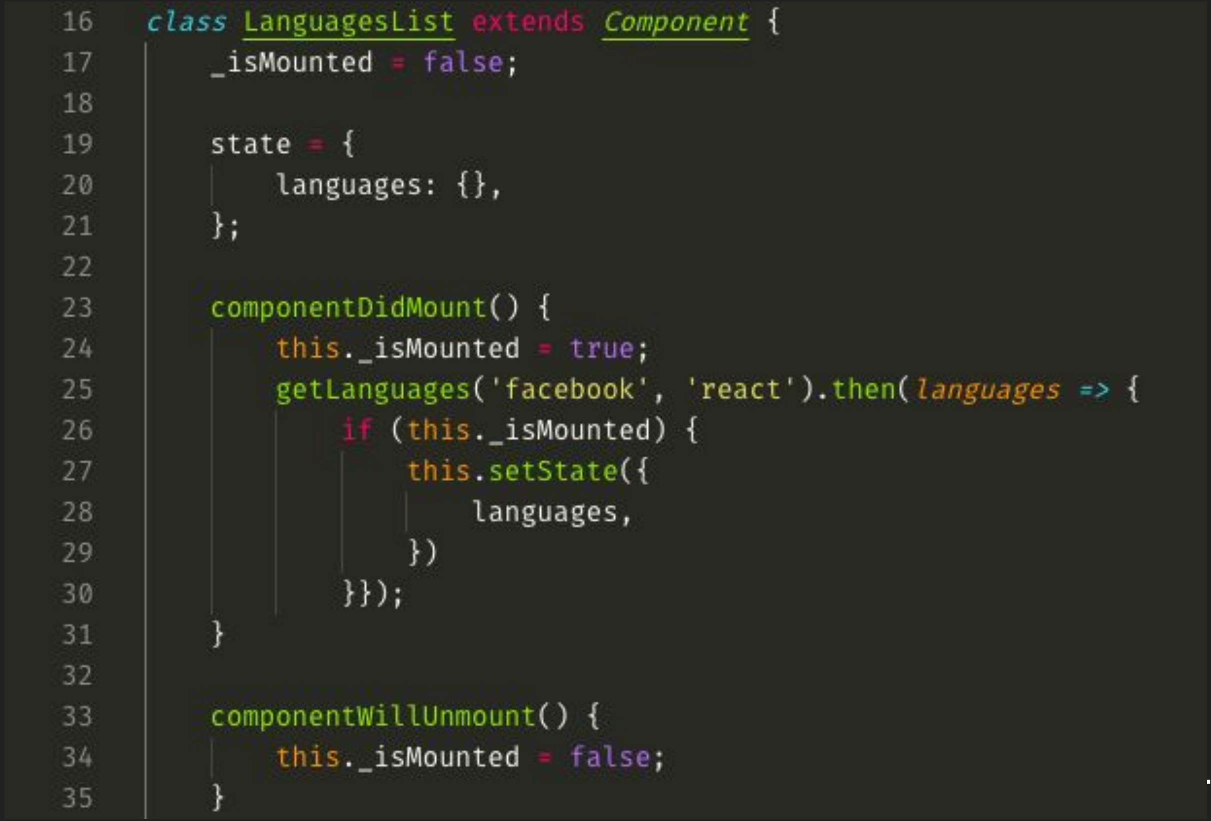
→ At this step, wait for a moment and see your test results. After finish the test, you will get __snapshots__/storybook.test.js.snap in src folder.

Possible warning: console.error /path/

Warning: Can't perform a React .... componentWillUnmount method.

in LanguagesList (at LanguagesList.stories.js:24)

If you want to fix,
fix by:



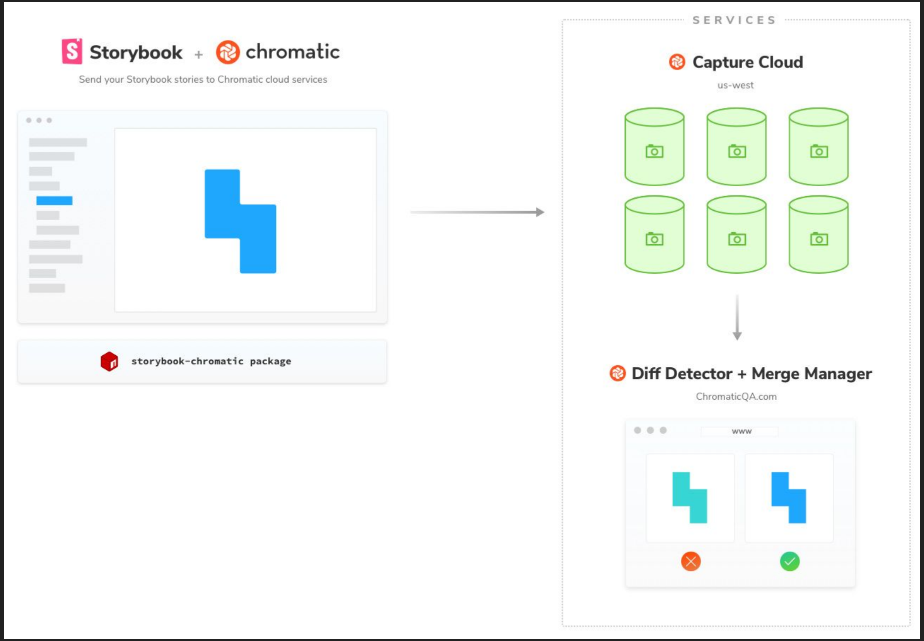
```
class LanguagesList extends Component {
    _isMounted = false;

    state = {
        languages: {},
    };

    componentDidMount() {
        this._isMounted = true;
        getLanguages('facebook', 'react').then(languages => {
            if (this._isMounted) {
                this.setState({
                    languages,
                })
            }});
    }

    componentWillUnmount() {
        this._isMounted = false;
    }
```

Testing our Storybook:

Type 2: Visual tests

Visual tests rely on developers to manually look at a component to verify it for correctness. They help us sanity check a component's appearance as we build.

Chromatic is a hassle-free Storybook addon for visual regression testing and review in the cloud.


Storybook + chromatic
Send your Storybook stories to Chromatic cloud services
storybook-chromatic package
SERVICES
Capture Cloud
us-west
Diff Detector + Merge Manager
ChromaticQA.com
www

1. Login via GitHub:

```
$ git add -A
$ git commit -m “UI test”
```

2. Install some devDependencies:

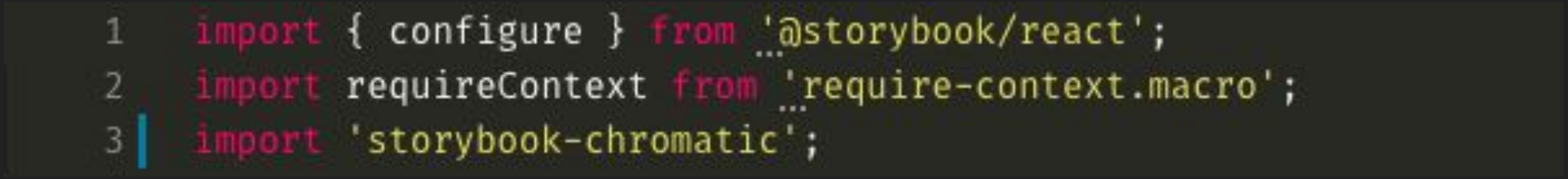
```
$ yarn add --dev react-chromatic storybook-chromatic
```

Note: You can also use

```
$ npm install react-chromatic storybook-chromatic --save-dev
```

3. Edit config.js file in '.storybook' folder:

# Go to .storybook/config.js
# Add line 3



```
1    import { configure } from '@storybook/react';
2    import requireContext from 'require-context.macro';
3    import 'storybook-chromatic';
```

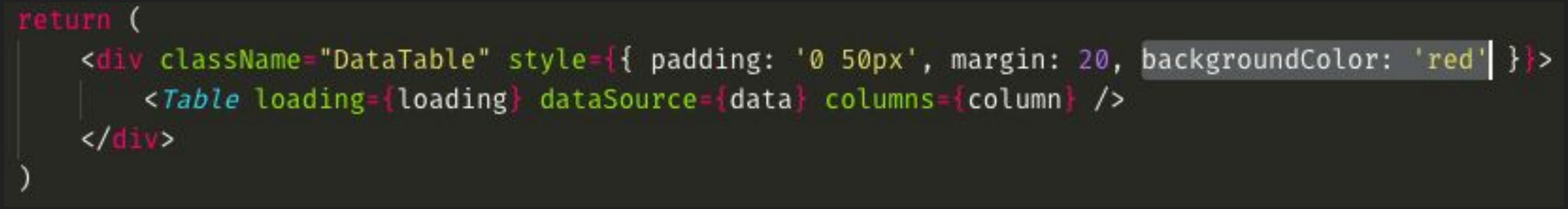
4. Go to https://www.chromaticqa.com/start then login with your GitHub and create a project with name "storyb".

5. Copy run command from the site and run in your Terminal:

```
$ ./node_modules/.bin/chromatic test --app-code=<app-code>
```

6. Catch a UI change:

```
# Go to src/DataTable.tsx and add backgroundColor: 'red'
```



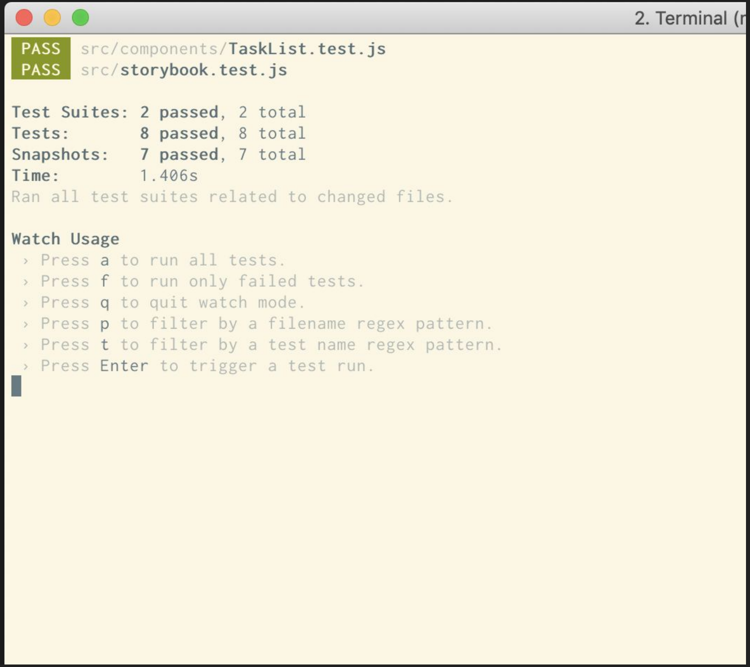
```
return (
    <div className="DataTable" style={{ padding: '0 50px', margin: 20, backgroundColor: 'red' }}>
        <Table loading={loading} dataSource={data} columns={column} />
    </div>
)
```

7. Run command in step 5 again and back to the web UI then you'll see changes.

Testing our Storybook:

Type 3: Unit tests

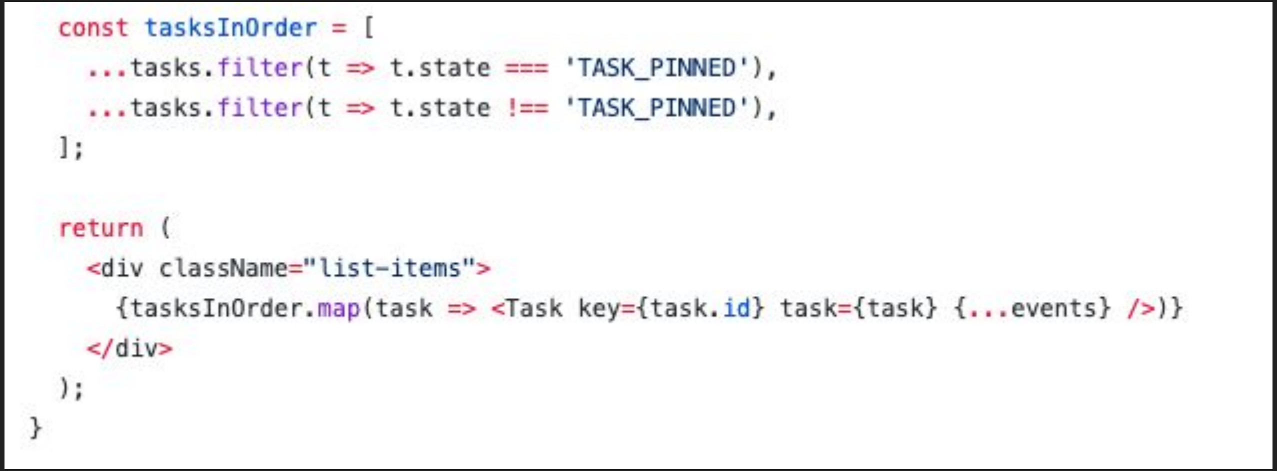
Unit tests with Jest verify that the output of a component remains the same given an fixed input. They're great for testing the functional qualities of a component.



```
PASS  src/components/TaskList.test.js
PASS  src/storybook.test.js

Test Suites: 2 passed, 2 total
Tests:       8 passed, 8 total
Snapshots:   7 passed, 7 total
Time:        1.406s
Ran all test suites related to changed files.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
```

Testing our Storybook:

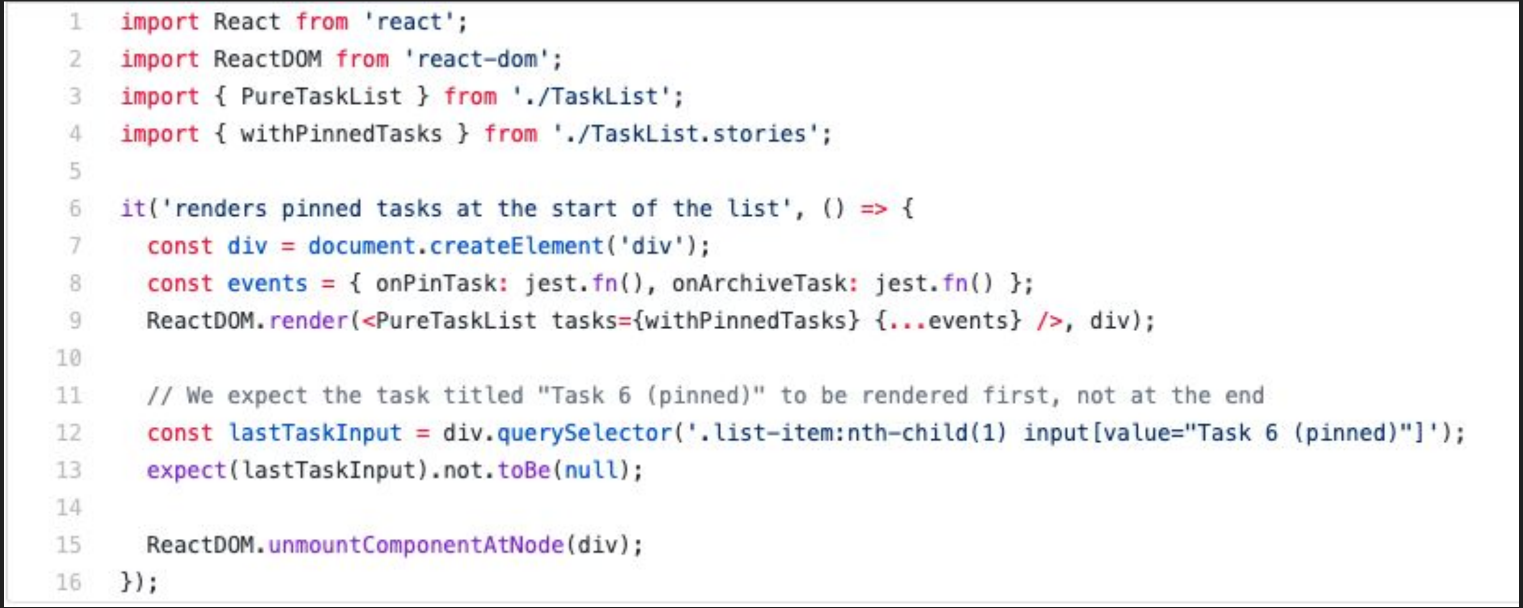
Type 3: Unit tests



```
  const tasksInOrder = [
    ...tasks.filter(t => t.state === 'TASK_PINNED'),
    ...tasks.filter(t => t.state !== 'TASK_PINNED'),
  ];

  return (
    <div className="list-items">
      {tasksInOrder.map(task => <Task key={task.id} task={task} {...events} />)}
    </div>
  );
}
```

Testing our Storybook:

Type 3: Unit tests



```
import React from 'react';
import ReactDOM from 'react-dom';
import { PureTaskList } from './TaskList';
import { withPinnedTasks } from './TaskList.stories';

it('renders pinned tasks at the start of the list', () => {
  const div = document.createElement('div');
  const events = { onPinTask: jest.fn(), onArchiveTask: jest.fn() };
  ReactDOM.render(<PureTaskList tasks={withPinnedTasks} {...events} />, div);

  // We expect the task titled "Task 6 (pinned)" to be rendered first, not at the end
  const lastTaskInput = div.querySelector('.list-item:nth-child(1) input[value="Task 6 (pinned)"]');
  expect(lastTaskInput).not.toBe(null);

  ReactDOM.unmountComponentAtNode(div);
});
```

References:

- Learn Storybook: https://www.learnstorybook.com/react/en/get-started/
- Storybook: https://storybook.js.org/docs/guides/guide-react/
- Chromatic: https://docs.chromaticqa.com/
- คิดจะทำ UI คิดถึง Storybook: https://bit.ly/2TKNmFq
- Visual unit testing with react-storybook and fetch-mock: https://bit.ly/31GvXTh