Scaling Team Delivery using Standardisation

Debbie O'Brien

Head Developer Advocate at Bit

Experts

mde

MVP Microsoft® Most Valuable Professional

debs_obrien

The Perfect Team

- Long term devs
- Work well together
- Know what you expect
- They do everything perfect

Reality

- Devs leave
- New devs join
- Devs get promoted and lead their own teams
- Teams get bigger as company scales

How do you

- keep up with constant team changes
- deliver software that looks like it was built by one team
- onboard new devs
- make sure they abide by standards

Write a 25 page booklet

- Do use CSS Modules
- Do write tests
- Do use Testing Library
- Do document your components
- Do add code examples

TECH

PLEASE FOLLOW INSTRUCTIONS

We need a solution

How?

- Know what you need from your team
- What standards you want
- Automate the process

Let's build a boilerplate

Boilerplate

Just clone the repo and get to work

Boilerplate

Just clone the repo and get to work

but

What happens when you want to update?
Maintaining a boilerplate project :(

Let's use Generators

With Generators

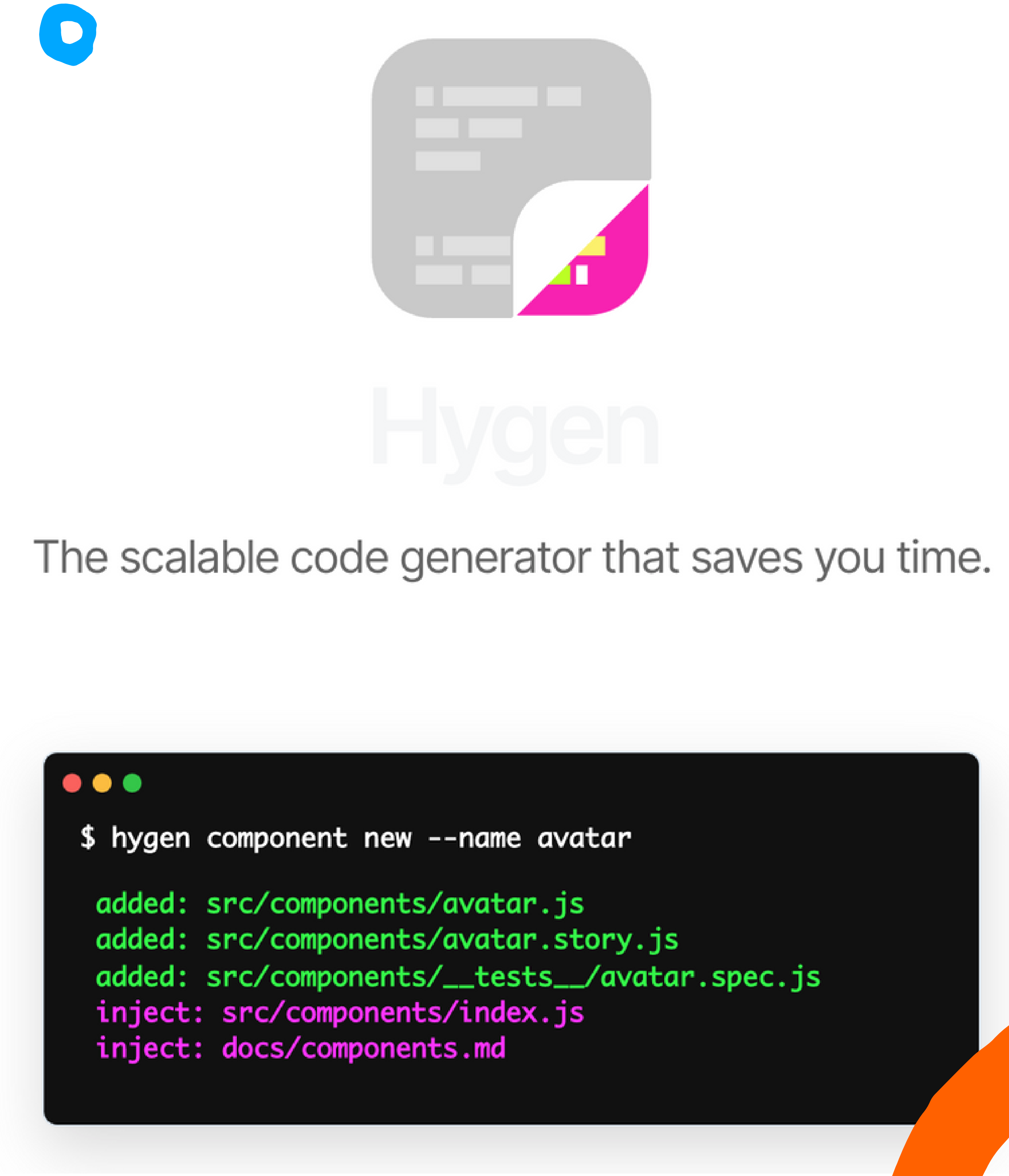
Generate code files set to the way you want

HOW?

Hygen

Header of a markdown-like frontmatter
Body of an ejs templating engine

Create new files
Inject into those files



Hygen

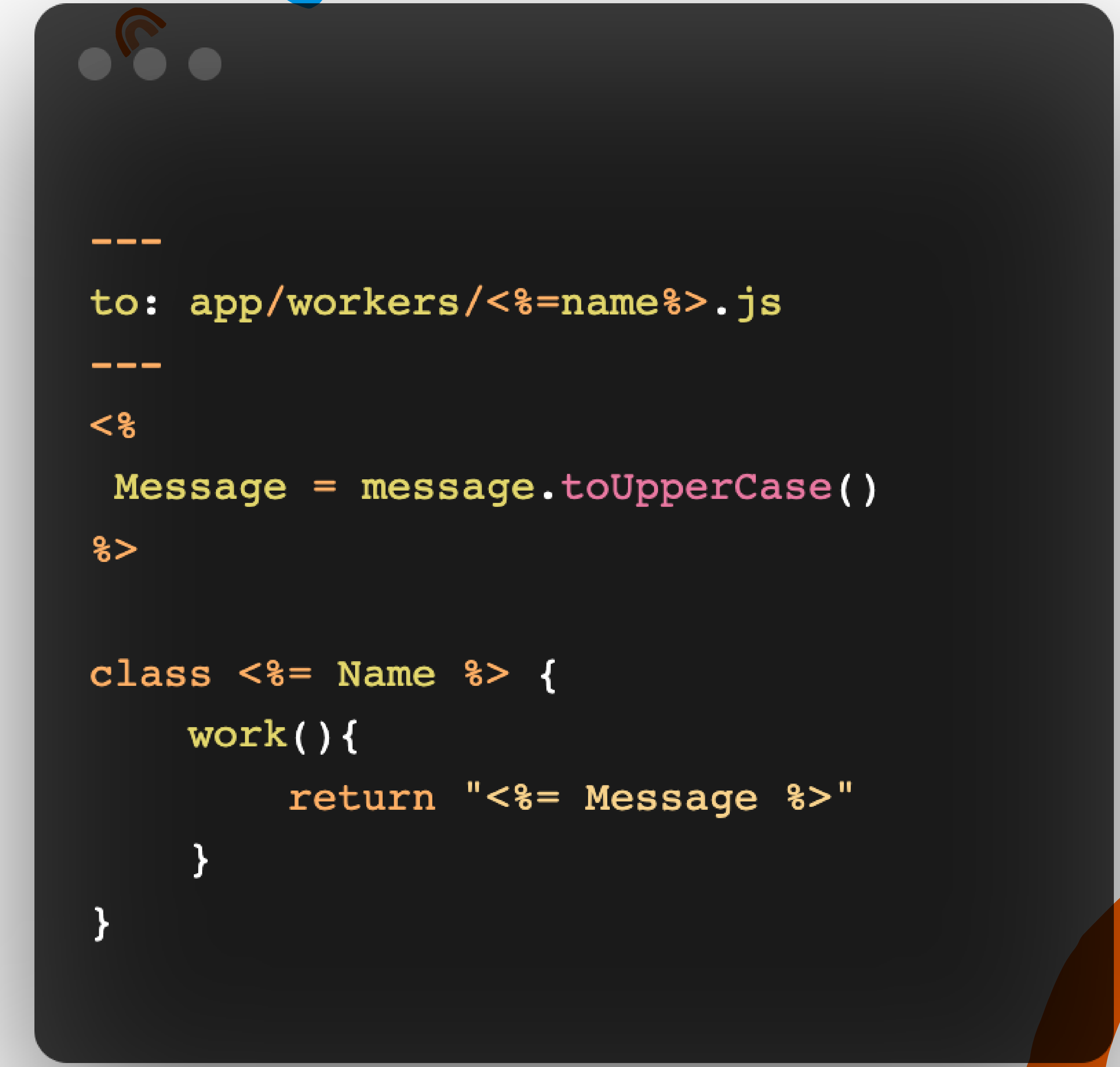
The scalable code generator that saves you time.

```
$ hygen component new --name avatar

  added: src/components/avatar.js
  added: src/components/avatar.story.js
  added: src/components/__tests__/avatar.spec.js
  inject: src/components/index.js
  inject: docs/components.md
```

Hygen

Header of a markdown-like frontmatter
Body of an ejs templating engine

Helpers included

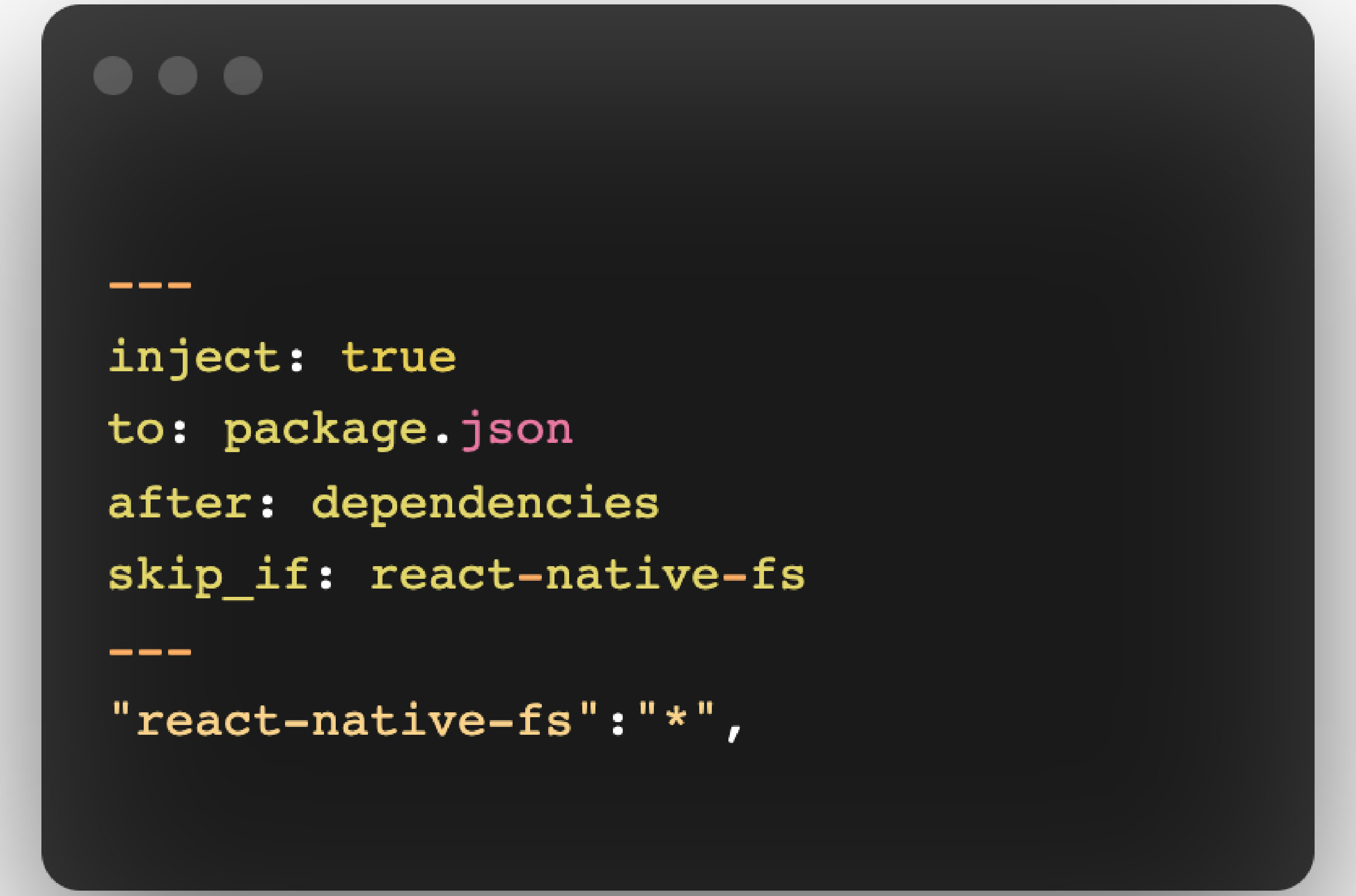


```
---
to: app/workers/<%=name%>.js
---
<%
 Message = message.toUpperCase()
%>

class <%= Name %> {
    work(){
        return "<%= Message %>"
    }
}
```

Hygen

Inject into files
add it after....

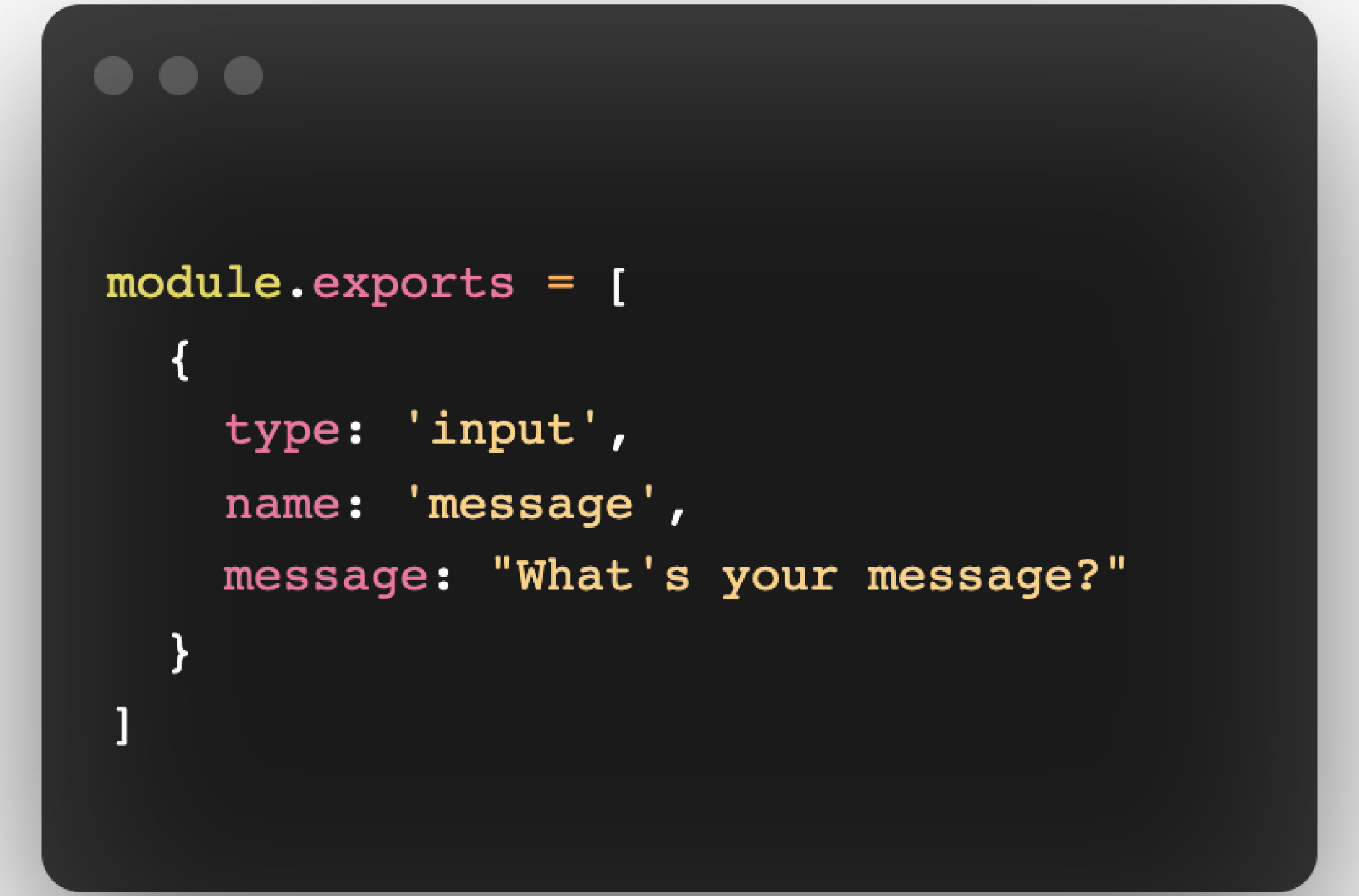
Skip if its already there



```
---
inject: true
to: package.json
after: dependencies
skip_if: react-native-fs
---
"react-native-fs":"*",
```

Hygen

Header of a markdown-like frontmatter
Body of an ejs templating engine



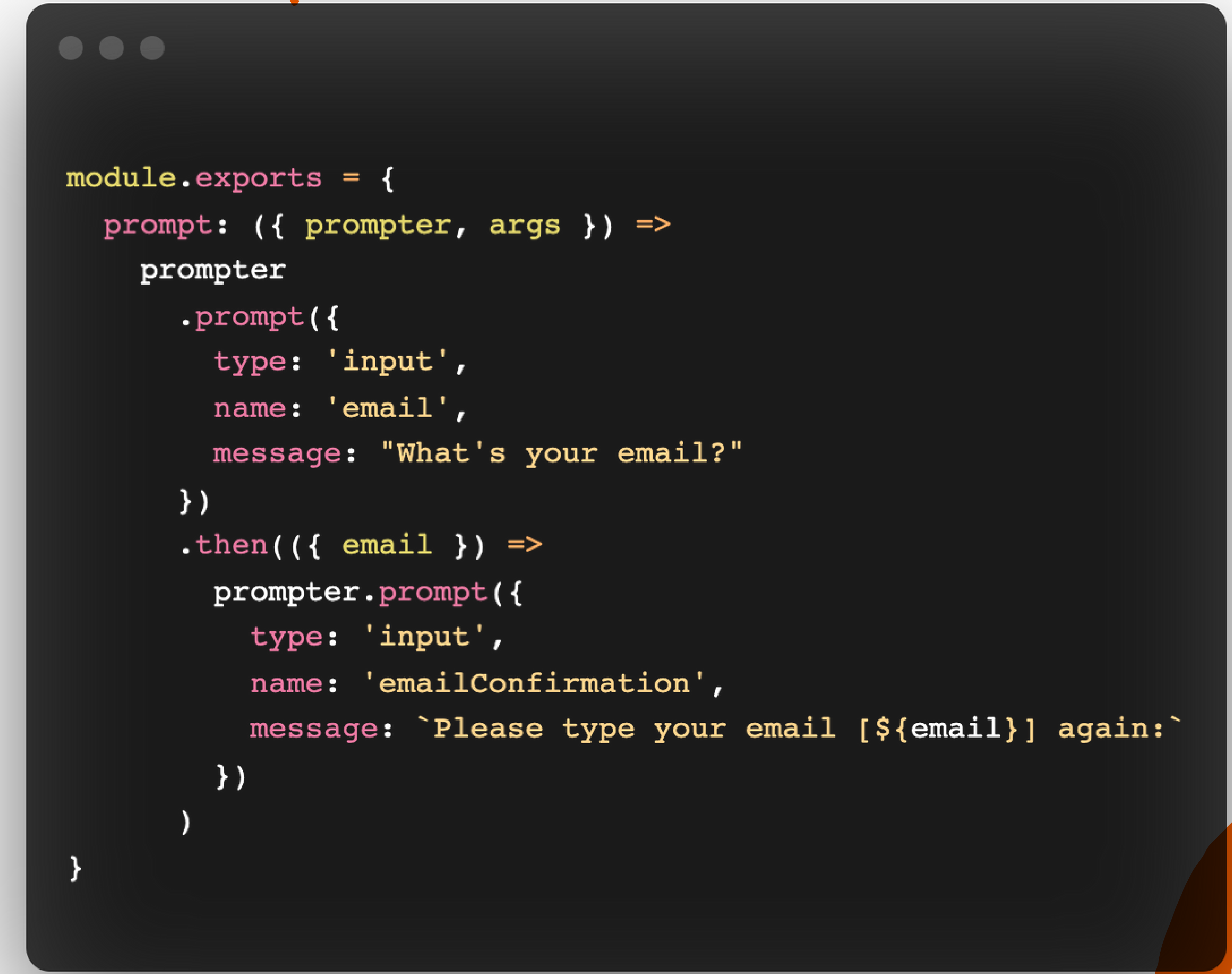
```
module.exports = [
  {
    type: 'input',
    name: 'message',
    message: "What's your message?"
  }
]
```

Interactive Prompts

Hygen

Header of a markdown-like frontmatter
Body of an ejs templating engine

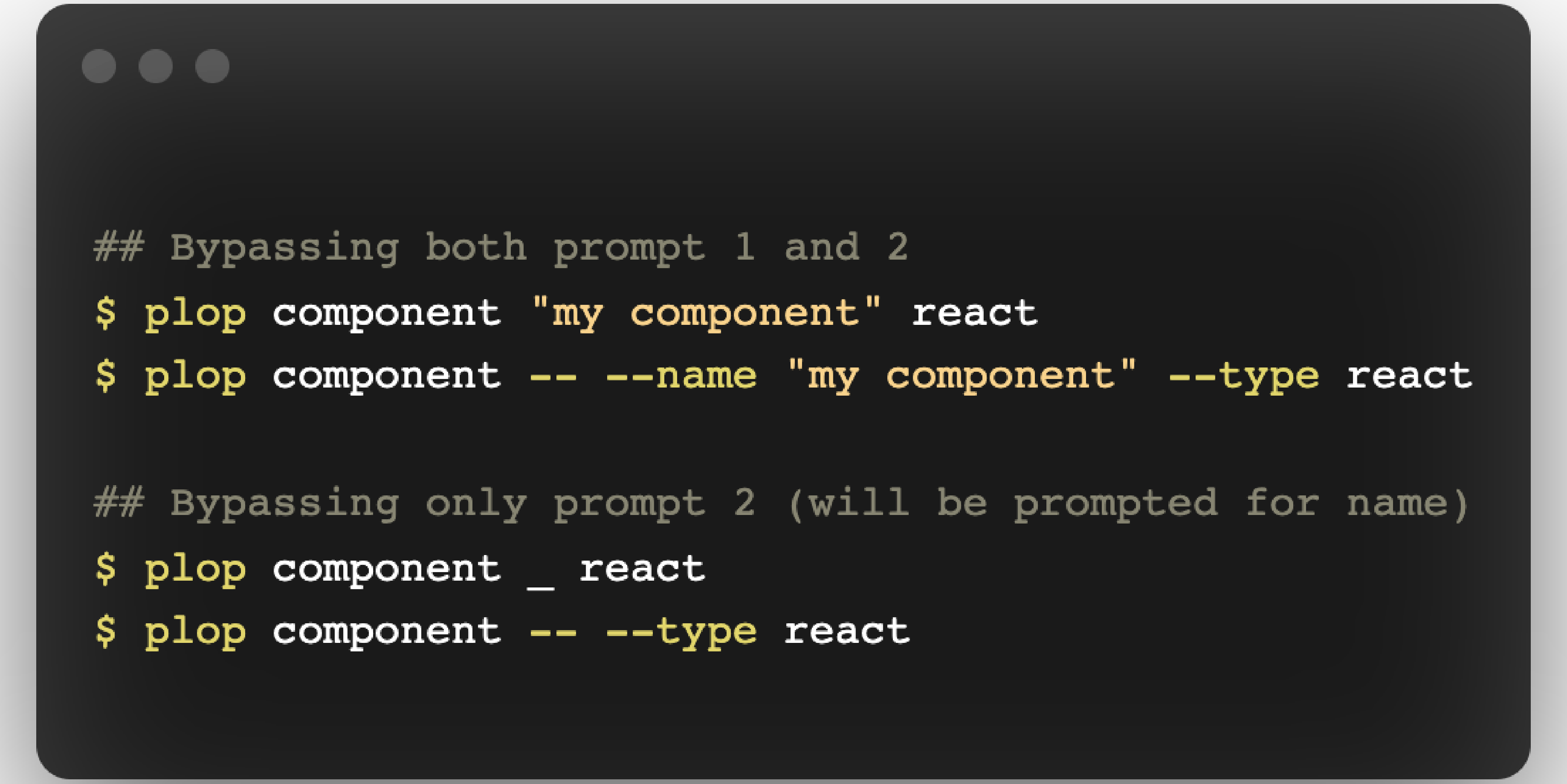
Interactive Prompts



```
module.exports = {
  prompt: ({ prompter, args }) =>
    prompter
      .prompt({
        type: 'input',
        name: 'email',
        message: "What's your email?"
      })
      .then(({ email }) =>
        prompter.prompt({
          type: 'input',
          name: 'emailConfirmation',
          message: `Please type your email [${email}] again:`
        })
      )
}
```

Plop

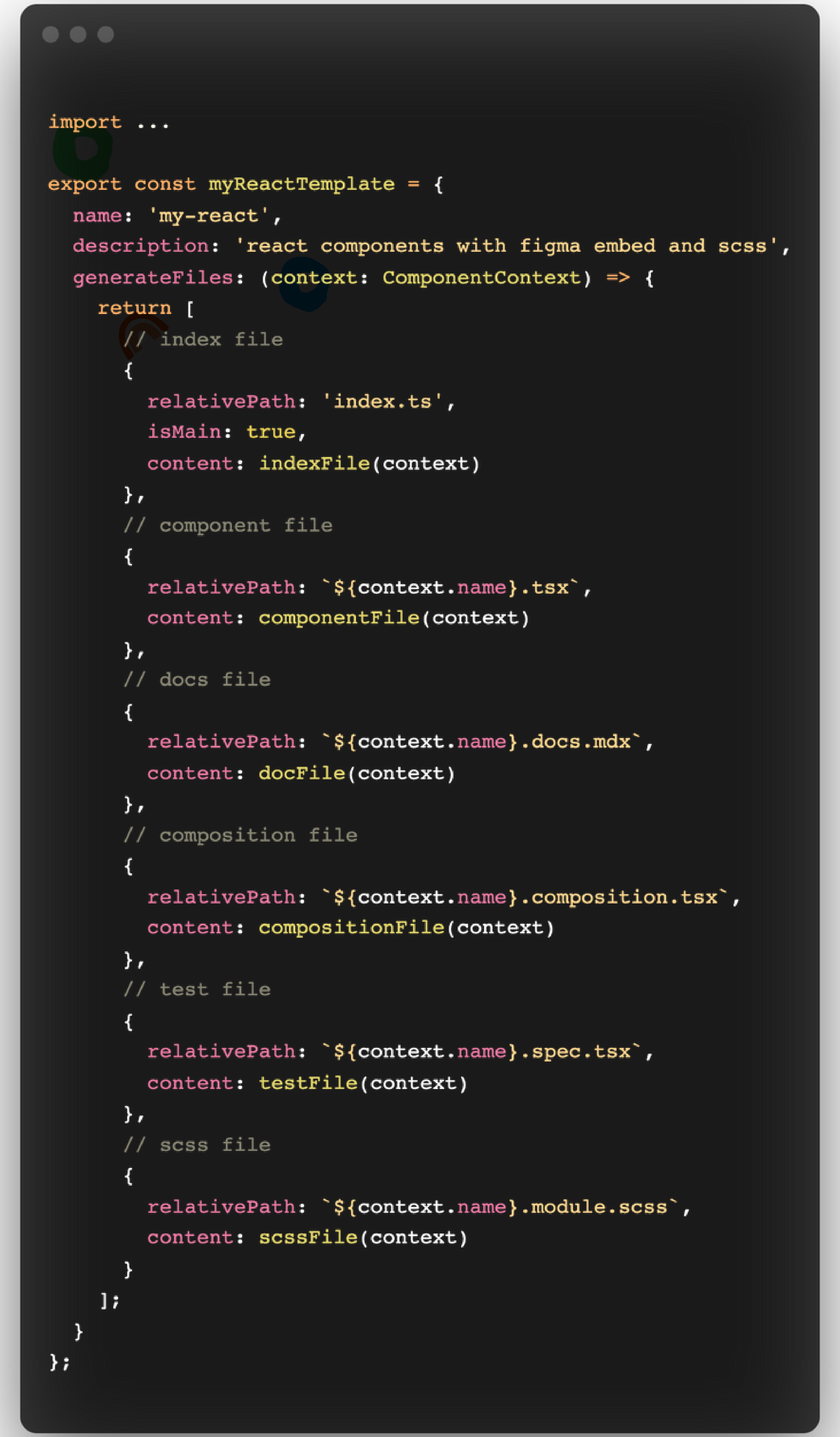
Inquirer Prompts
Handlebar templates



```
## Bypassing both prompt 1 and 2
$ plop component "my component" react
$ plop component -- --name "my component" --type react

## Bypassing only prompt 2 (will be prompted for name)
$ plop component _ react
$ plop component -- --type react
```

Bit

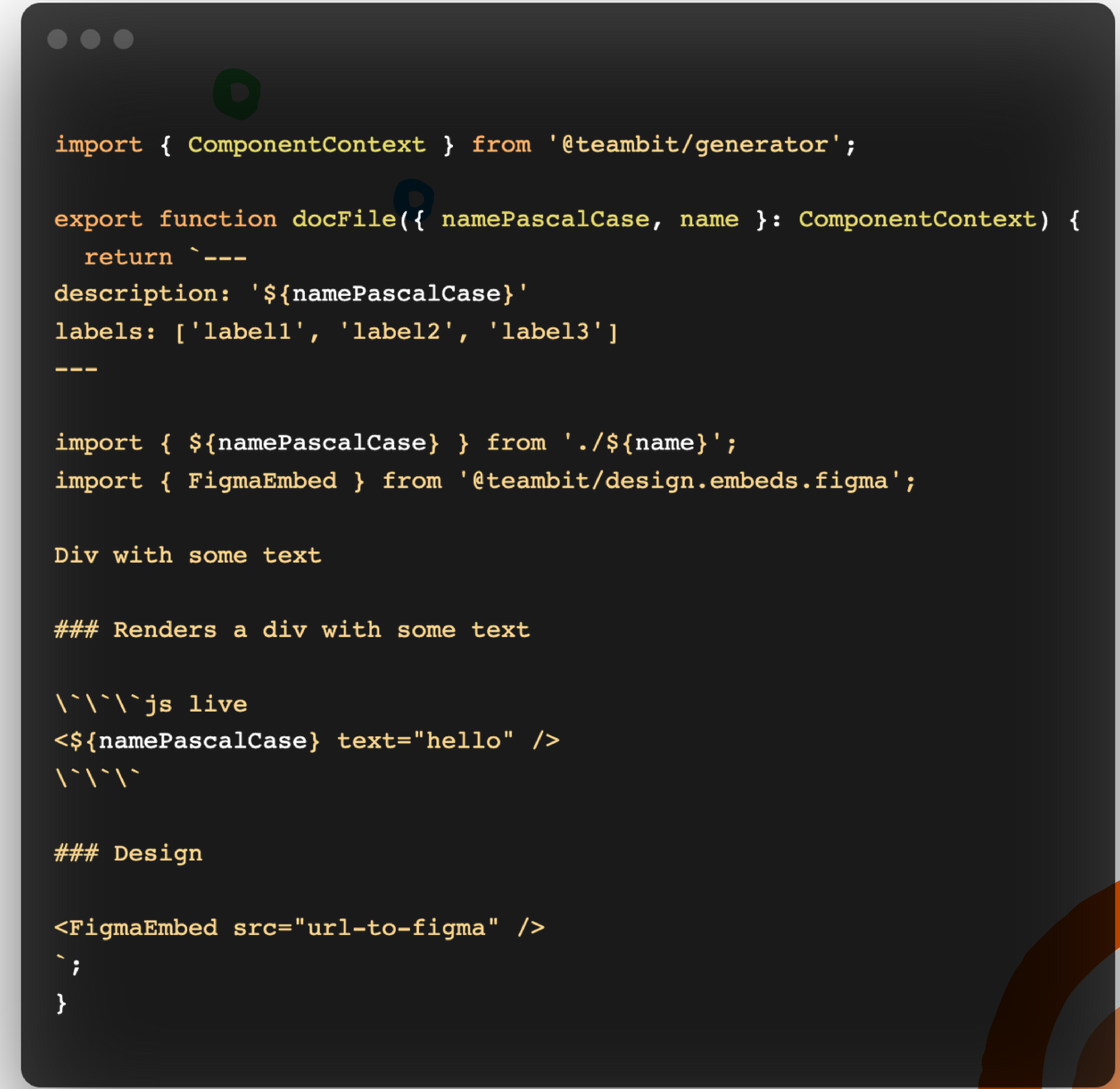
Component Generator
Generate Multiple files



```
import ...

export const myReactTemplate = {
  name: 'my-react',
  description: 'react components with figma embed and scss',
  generateFiles: (context: ComponentContext) => {
    return [
      // index file
      {
        relativePath: 'index.ts',
        isMain: true,
        content: indexFile(context)
      },
      // component file
      {
        relativePath: `${context.name}.tsx`,
        content: componentFile(context)
      },
      // docs file
      {
        relativePath: `${context.name}.docs.mdx`,
        content: docFile(context)
      },
      // composition file
      {
        relativePath: `${context.name}.composition.tsx`,
        content: compositionFile(context)
      },
      // test file
      {
        relativePath: `${context.name}.spec.tsx`,
        content: testFile(context)
      },
      // scss file
      {
        relativePath: `${context.name}.module.scss`,
        content: scssFile(context)
      }
    ];
  }
};
```

Bit

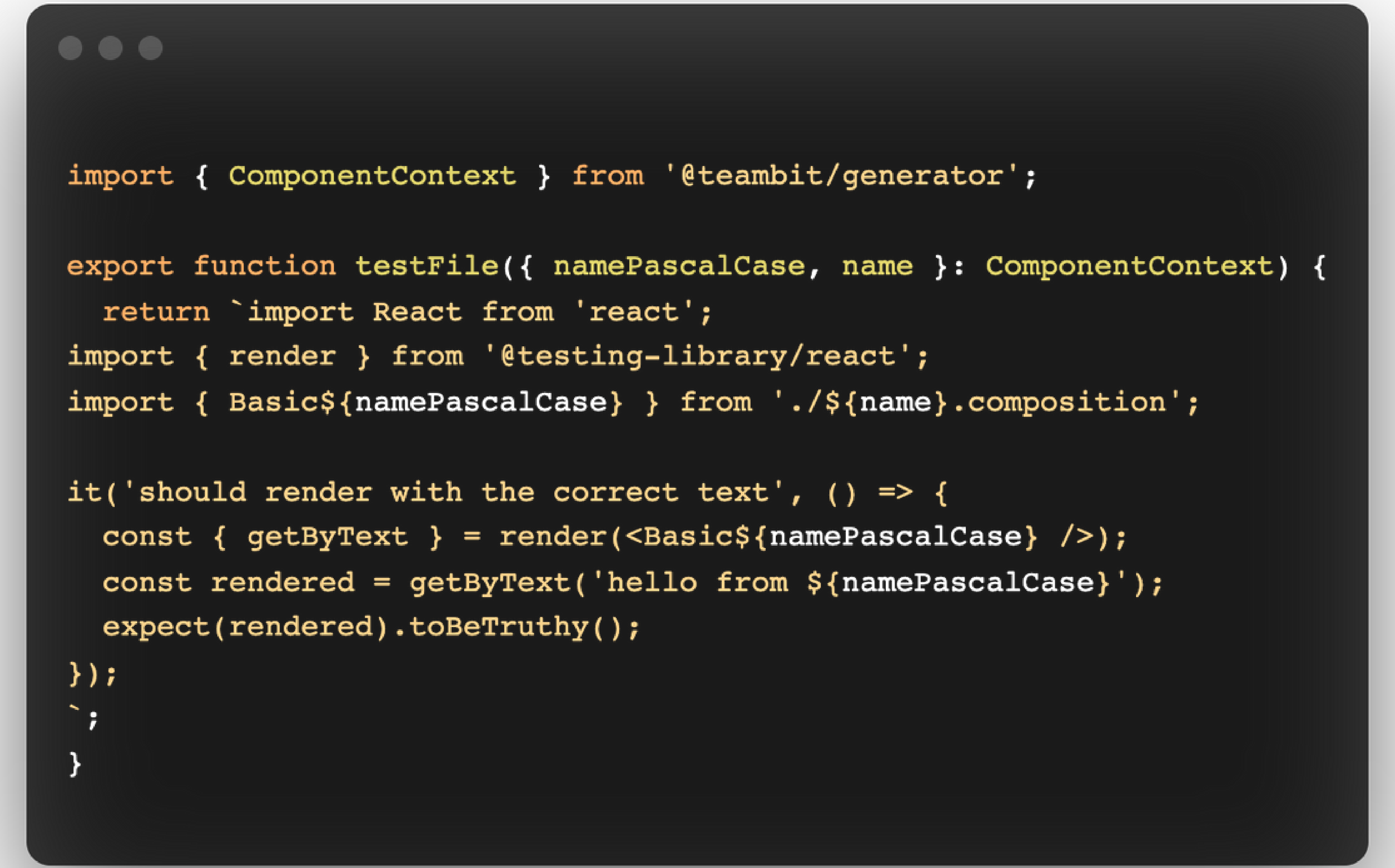
Render docs for your component
Easier onboarding for new devs



```
import { ComponentContext } from '@teambit/generator';

export function docFile({ namePascalCase, name }: ComponentContext) {
  return `---
description: '${namePascalCase}'
labels: ['label1', 'label2', 'label3']
---

import { ${namePascalCase} } from './${name}';
import { FigmaEmbed } from '@teambit/design.embeds.figma';

Div with some text

### Renders a div with some text

\`\`\`js live
<${namePascalCase} text="hello" />
\`\`\`

### Design

<FigmaEmbed src="url-to-figma" />
`;
}
```

Bit

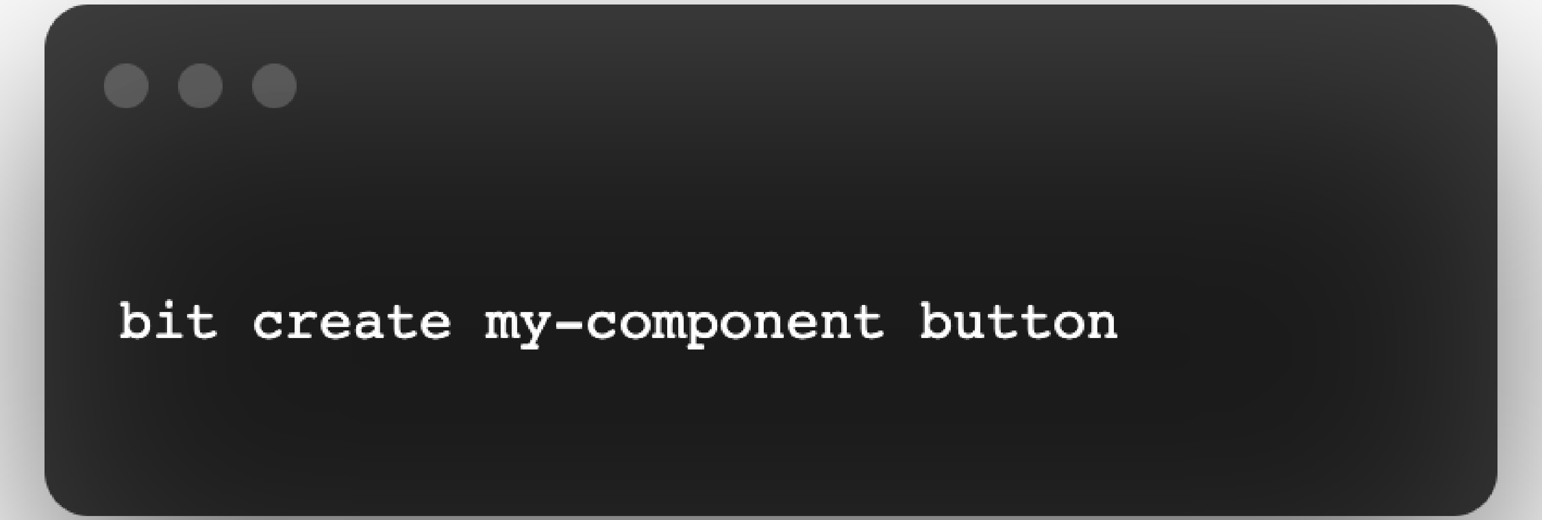
Render tests for your component
Easier onboarding for new devs



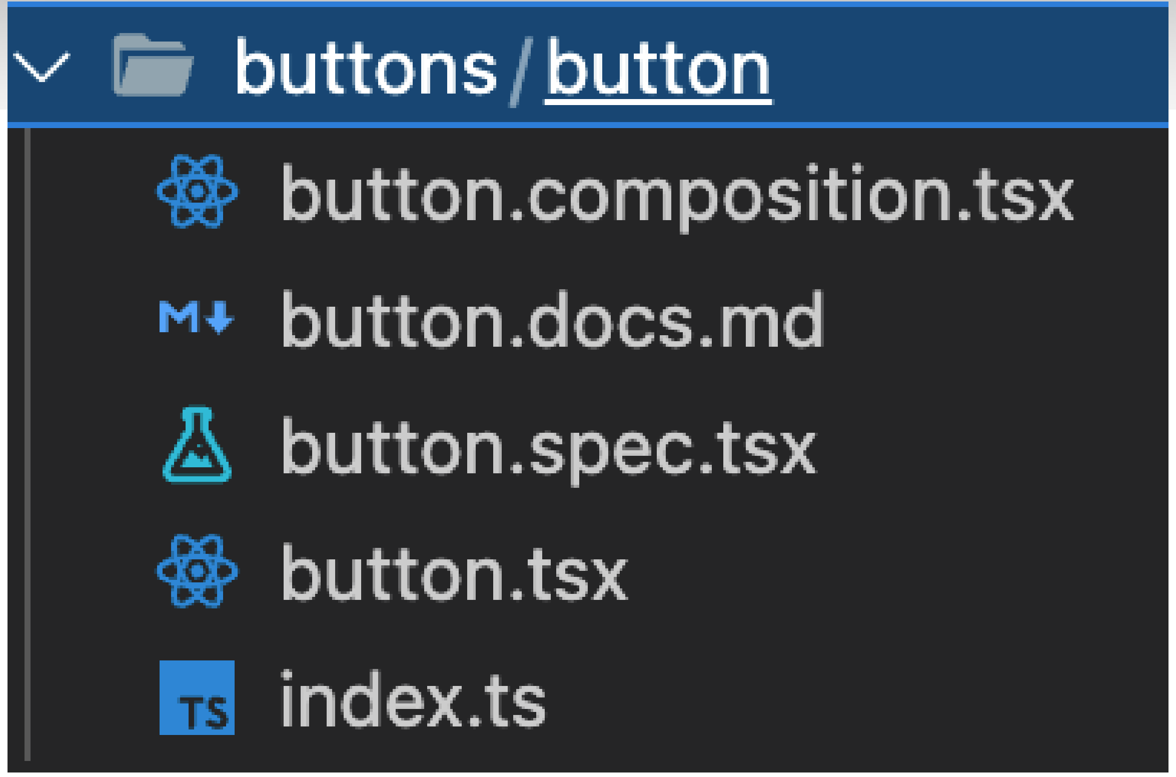
```
import { ComponentContext } from '@teambit/generator';

export function testFile({ namePascalCase, name }: ComponentContext) {
  return `import React from 'react';
import { render } from '@testing-library/react';
import { Basic${namePascalCase} } from './${name}.composition';

it('should render with the correct text', () => {
  const { getByText } = render(<Basic${namePascalCase} />);
  const rendered = getByText('hello from ${namePascalCase}');
  expect(rendered).toBeTruthy();
});
`;
}
```

Bit

Using your generator



```
bit create my-component button
```



```
buttons/button
    button.composition.tsx
    button.docs.md
    button.spec.tsx
    button.tsx
    index.ts
```

Isn't it a lot of work?

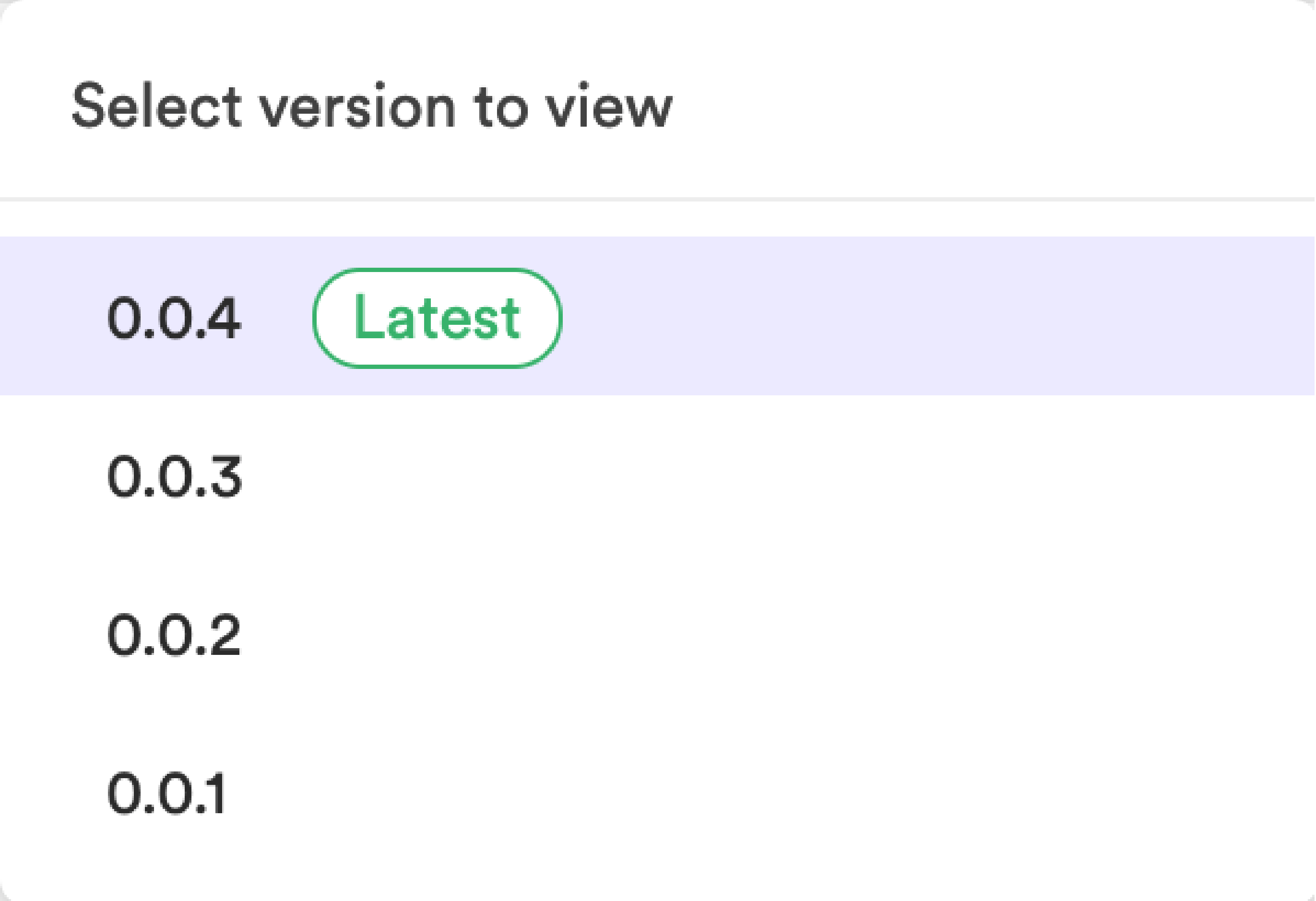
- Create once, use many times
- Save 5 mins today, tomorrow...
- Easier onboarding
- Ensure standards
- Devs concentrate on what's important

What if we need to make changes?

Bit

Your generator is a component
Independent versions

Make a change, create a new version



Select version to view

0.0.4 Latest

0.0.3

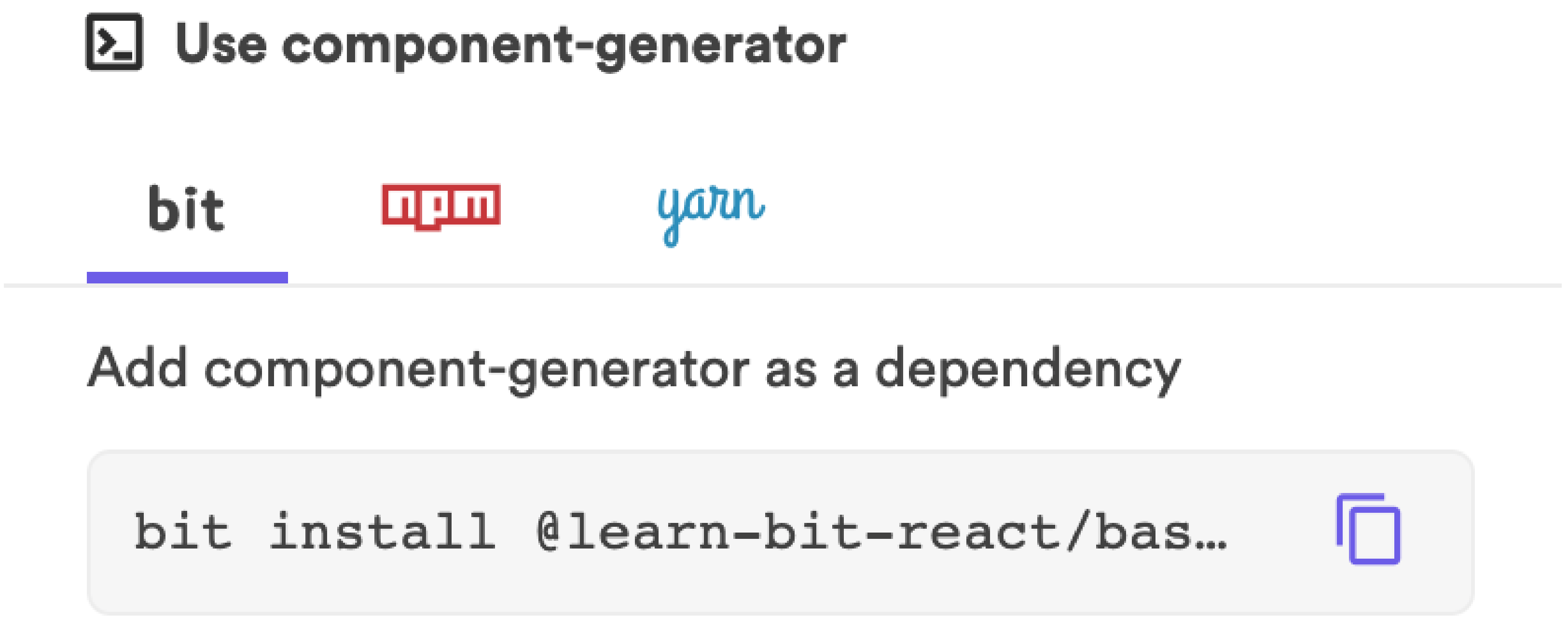
0.0.2

0.0.1

Bit

Can be installed and used in any project/app

Choose which version to install



Use component-generator

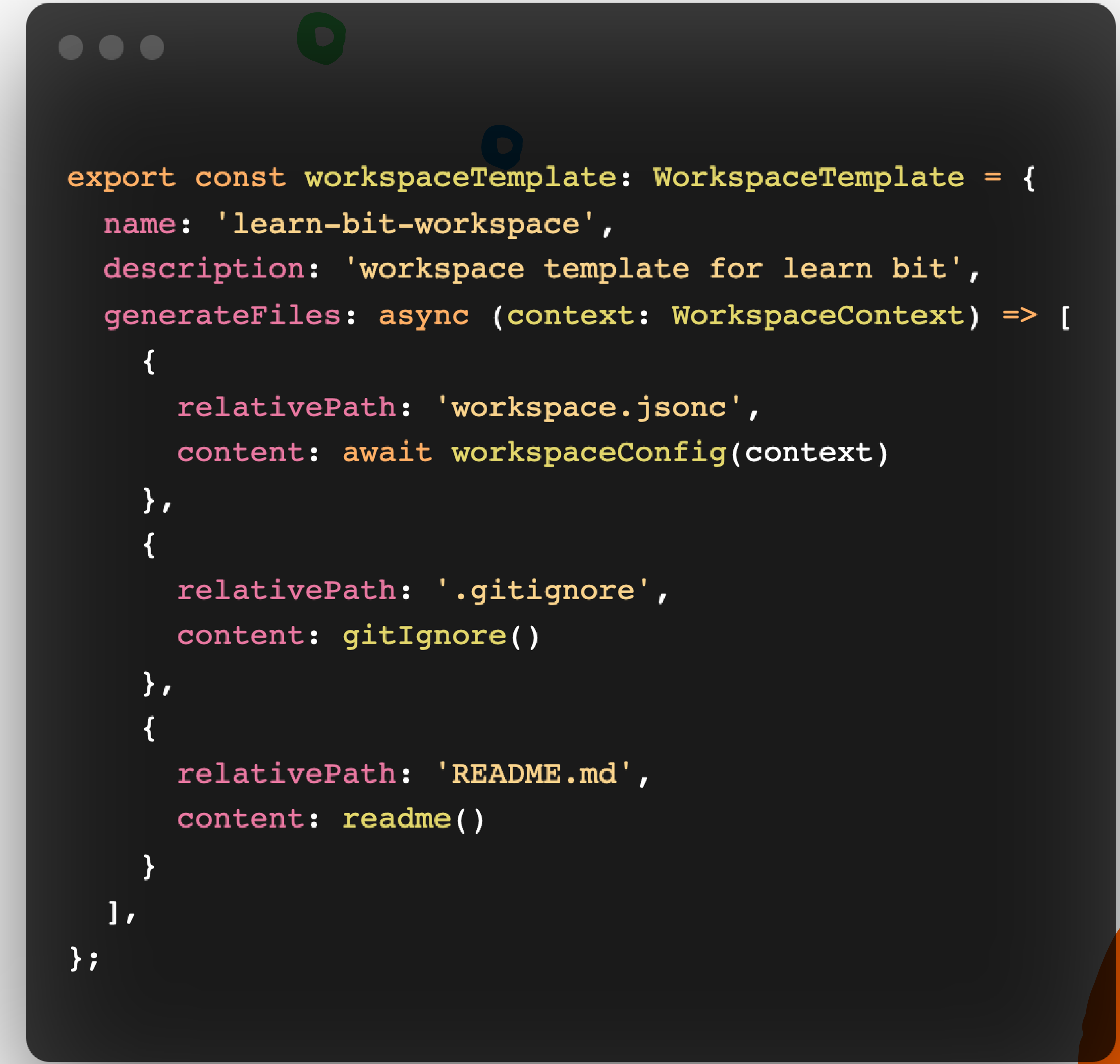
Add component-generator as a dependency

```
bit install @learn-bit-react/bas…
```

What about starting new projects?

Bit

Generate a workspace
Generate files

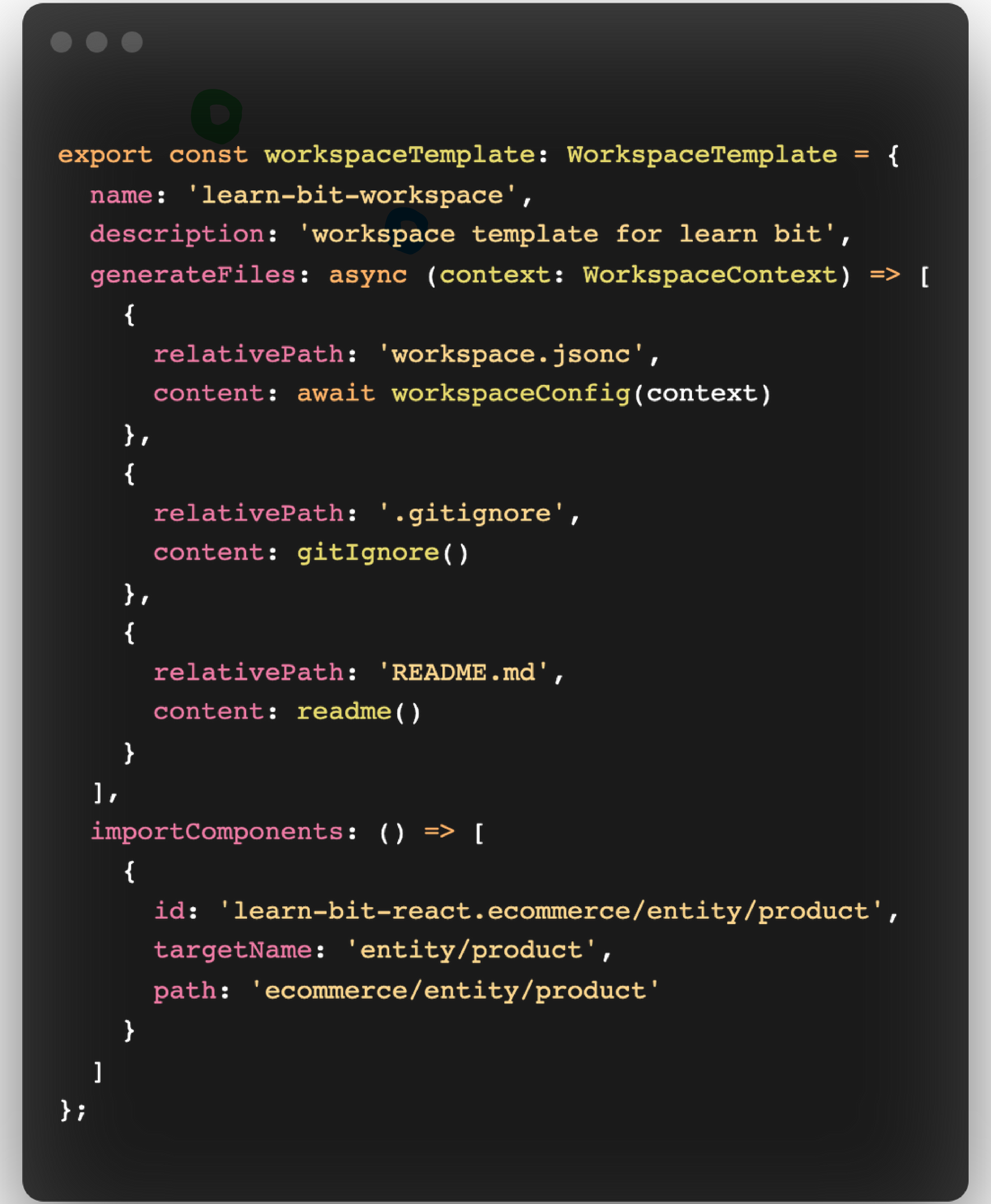


```
export const workspaceTemplate: WorkspaceTemplate = {
  name: 'learn-bit-workspace',
  description: 'workspace template for learn bit',
  generateFiles: async (context: WorkspaceContext) => [
    {
      relativePath: 'workspace.jsonc',
      content: await workspaceConfig(context)
    },
    {
      relativePath: '.gitignore',
      content: gitIgnore()
    },
    {
      relativePath: 'README.md',
      content: readme()
    }
  ],
};
```

Bit

Generate a workspace
Generate files

Add components



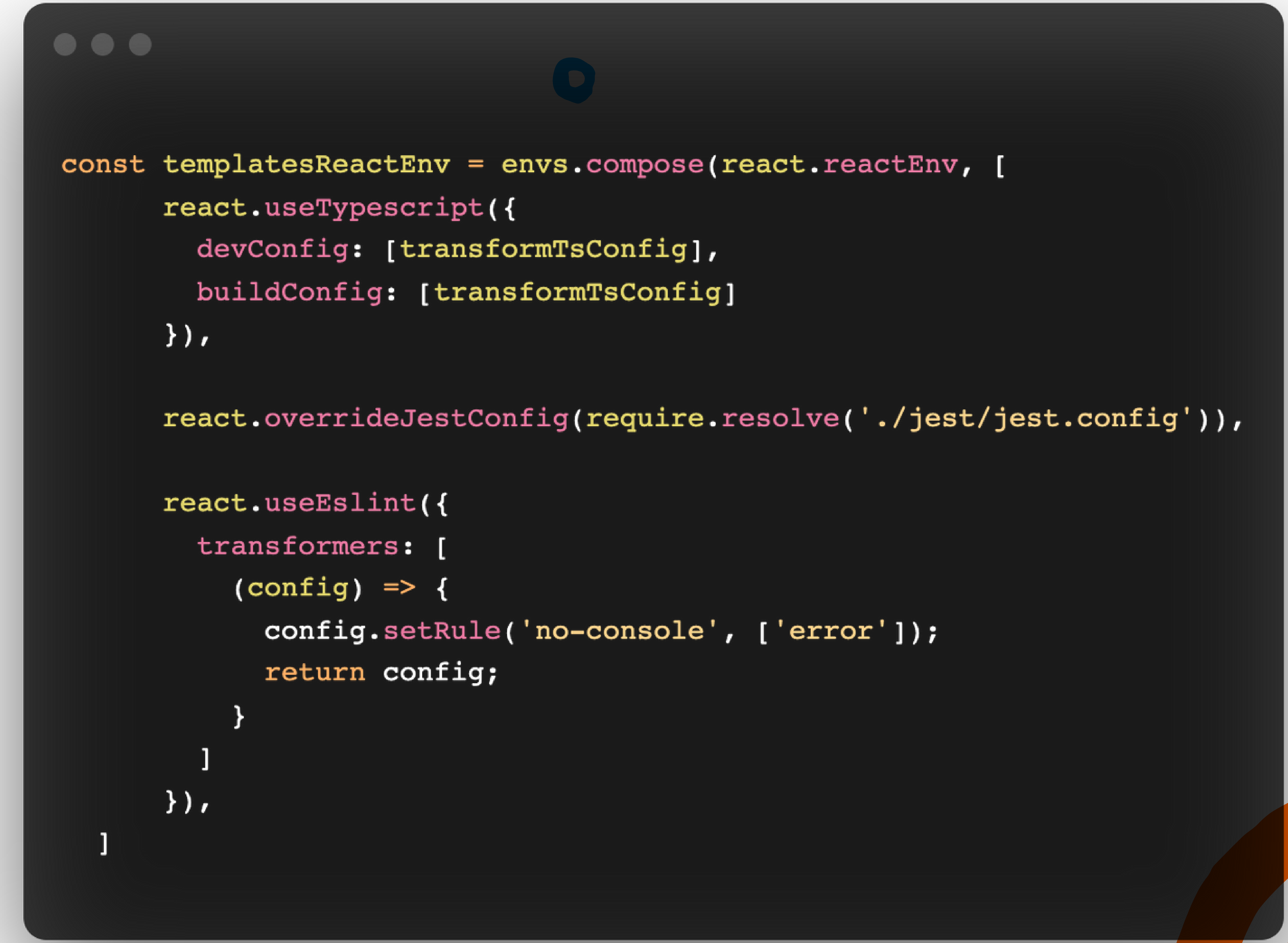
```
export const workspaceTemplate: WorkspaceTemplate = {
  name: 'learn-bit-workspace',
  description: 'workspace template for learn bit',
  generateFiles: async (context: WorkspaceContext) => [
    {
      relativePath: 'workspace.jsonc',
      content: await workspaceConfig(context)
    },
    {
      relativePath: '.gitignore',
      content: gitIgnore()
    },
    {
      relativePath: 'README.md',
      content: readme()
    }
  ],
  importComponents: () => [
    {
      id: 'learn-bit-react.ecommerce/entity/product',
      targetName: 'entity/product',
      path: 'ecommerce/entity/product'
    }
  ]
};
```

What about configs?

Bit

Create an environment
It's just a component

One config for all components

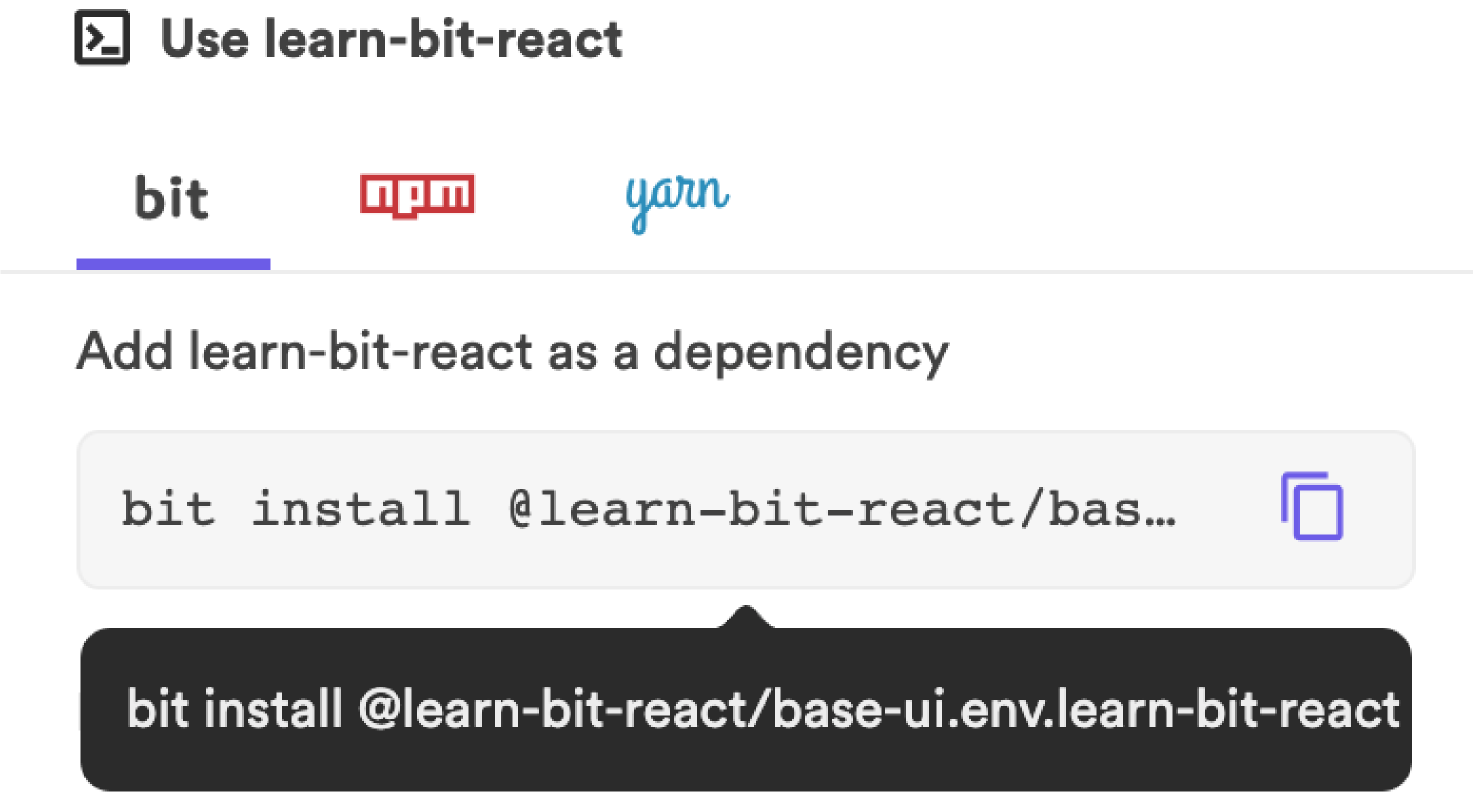


```
const templatesReactEnv = envs.compose(react.reactEnv, [
   react.useTypescript({
    devConfig: [transformTsConfig],
    buildConfig: [transformTsConfig]
   }),

   react.overrideJestConfig(require.resolve('./jest/jest.config')),

   react.useEslint({
    transformers: [
     (config) => {
      config.setRule('no-console', ['error']);
      return config;
     }
    ]
   }),
 ]
```

Bit

Install it to any workspace
Apply it to all components

Make a change
Create a new version



Use learn-bit-react

Add learn-bit-react as a dependency

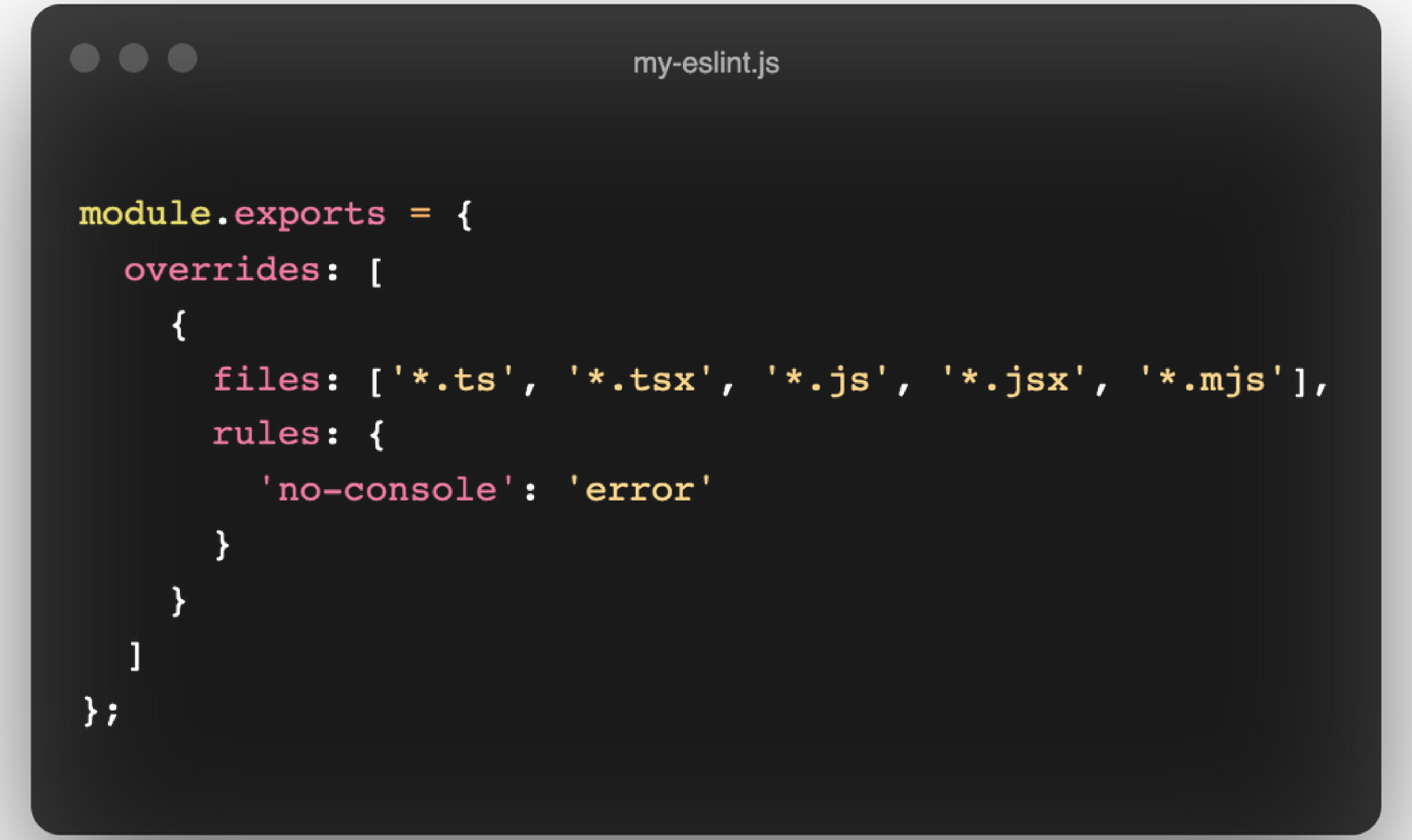
```
bit install @learn-bit-react/bas…
```

```
bit install @learn-bit-react/base-ui.env.learn-bit-react
```

Configs for non Bit Projects

Sharing ESLint config

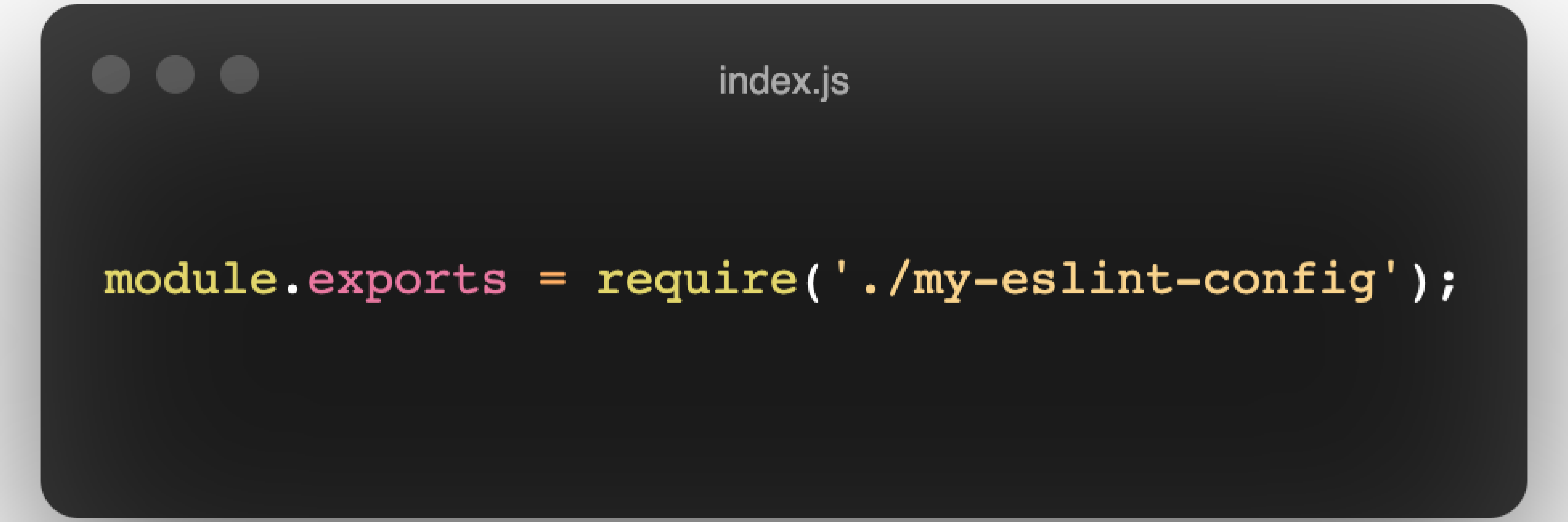
Create your config as a component
Add your rules



```
module.exports = {
  overrides: [
    {
      files: ['*.ts', '*.tsx', '*.js', '*.jsx', '*.mjs'],
      rules: {
        'no-console': 'error'
      }
    }
  ]
};
```

Sharing ESLint config

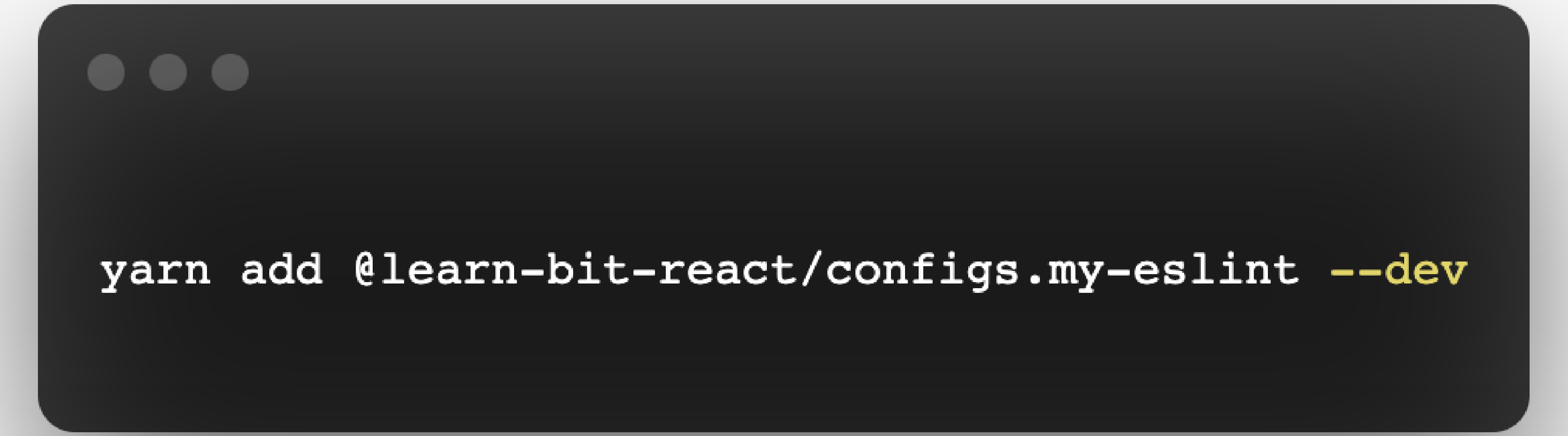
Export the config file



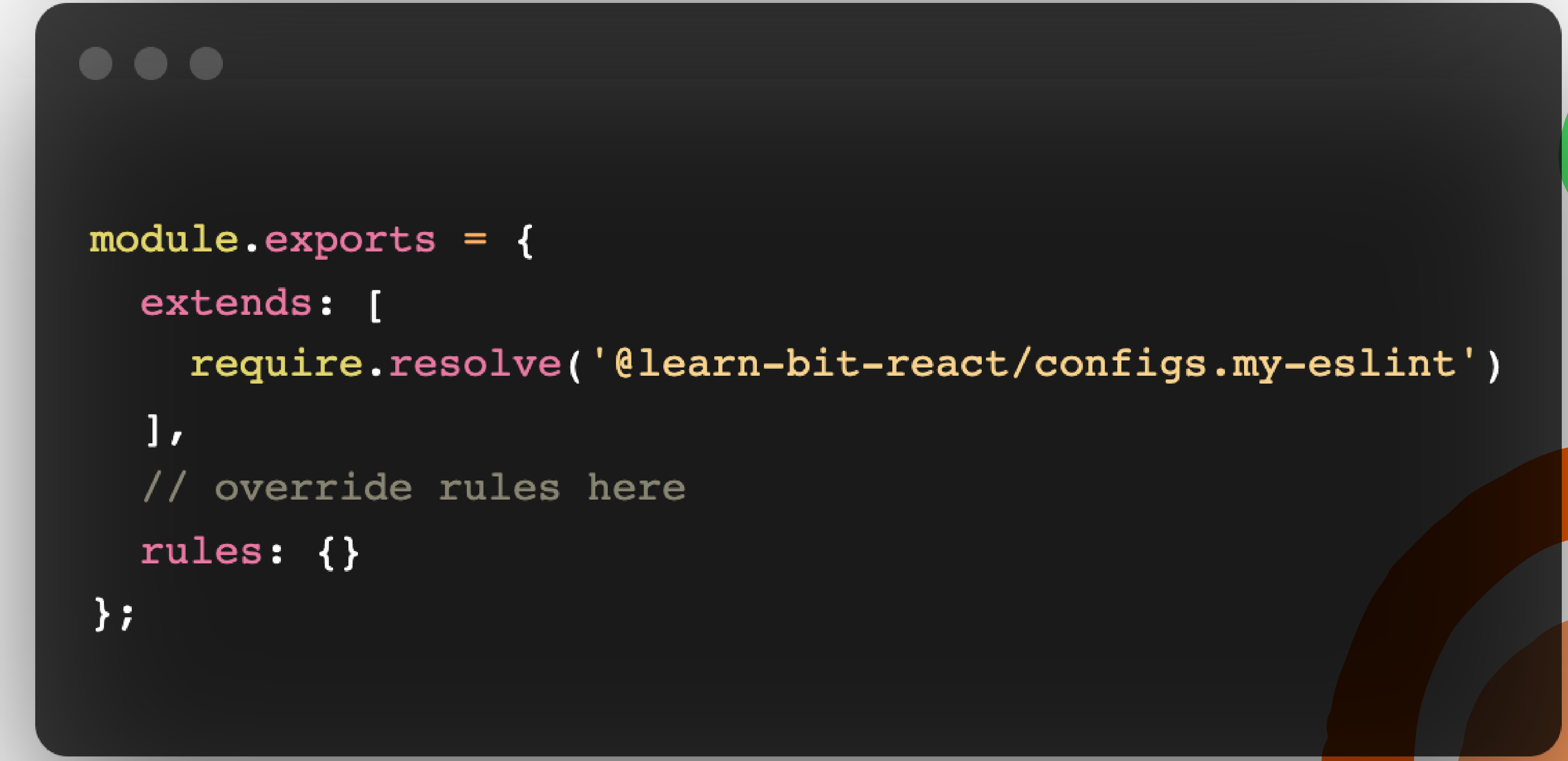
```
module.exports = require('./my-eslint-config');
```

Sharing ESLint config

Import the config component
All projects stay up to date



```
yarn add @learn-bit-react/configs.my-eslint --dev
```

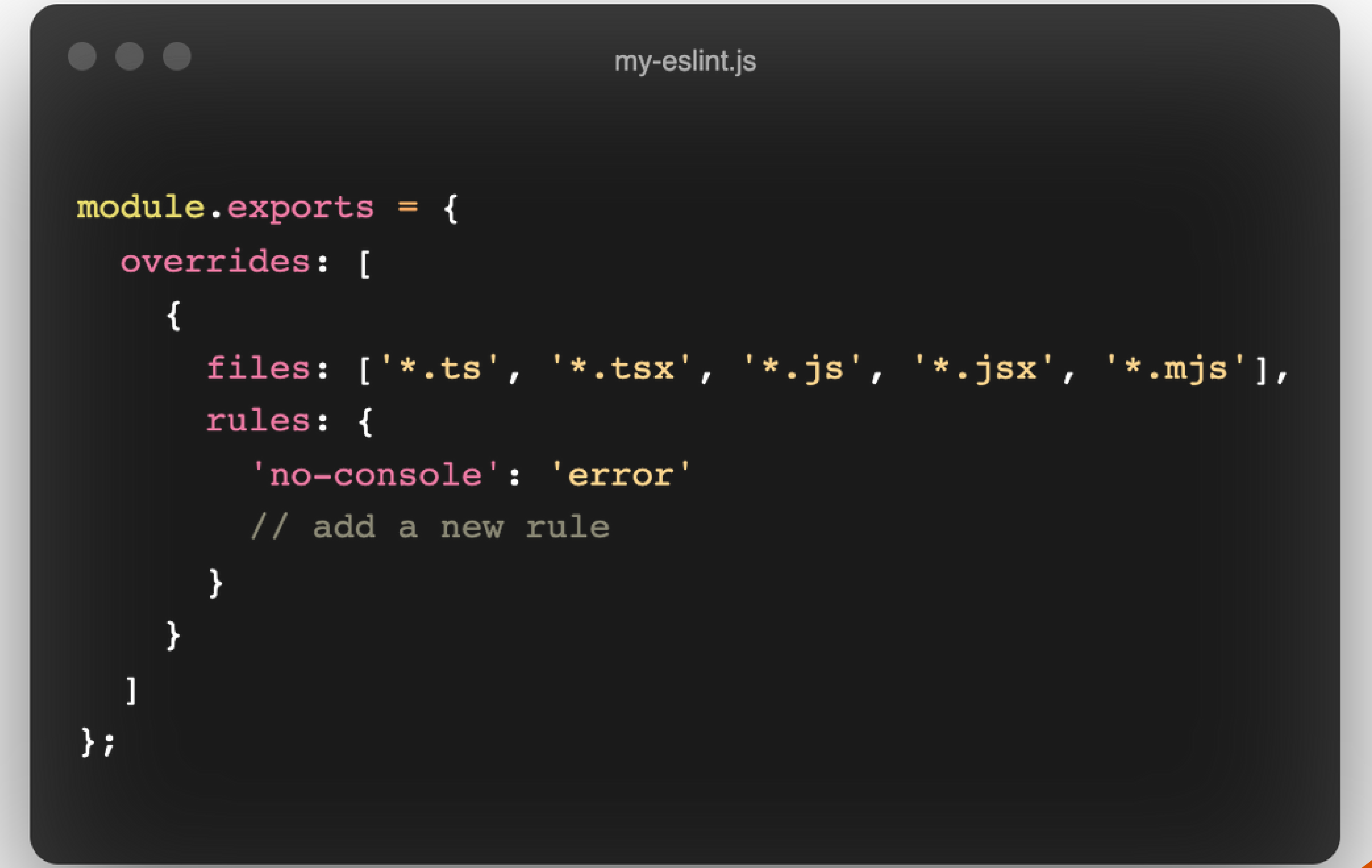


```
module.exports = {
  extends: [
    require.resolve('@learn-bit-react/configs.my-eslint')
  ],
  // override rules here
  rules: {}
};
```

Sharing ESLint config

Add a new rule or make a change
Create a new version of your config component

Update the version in your projects
All projects stay up to date

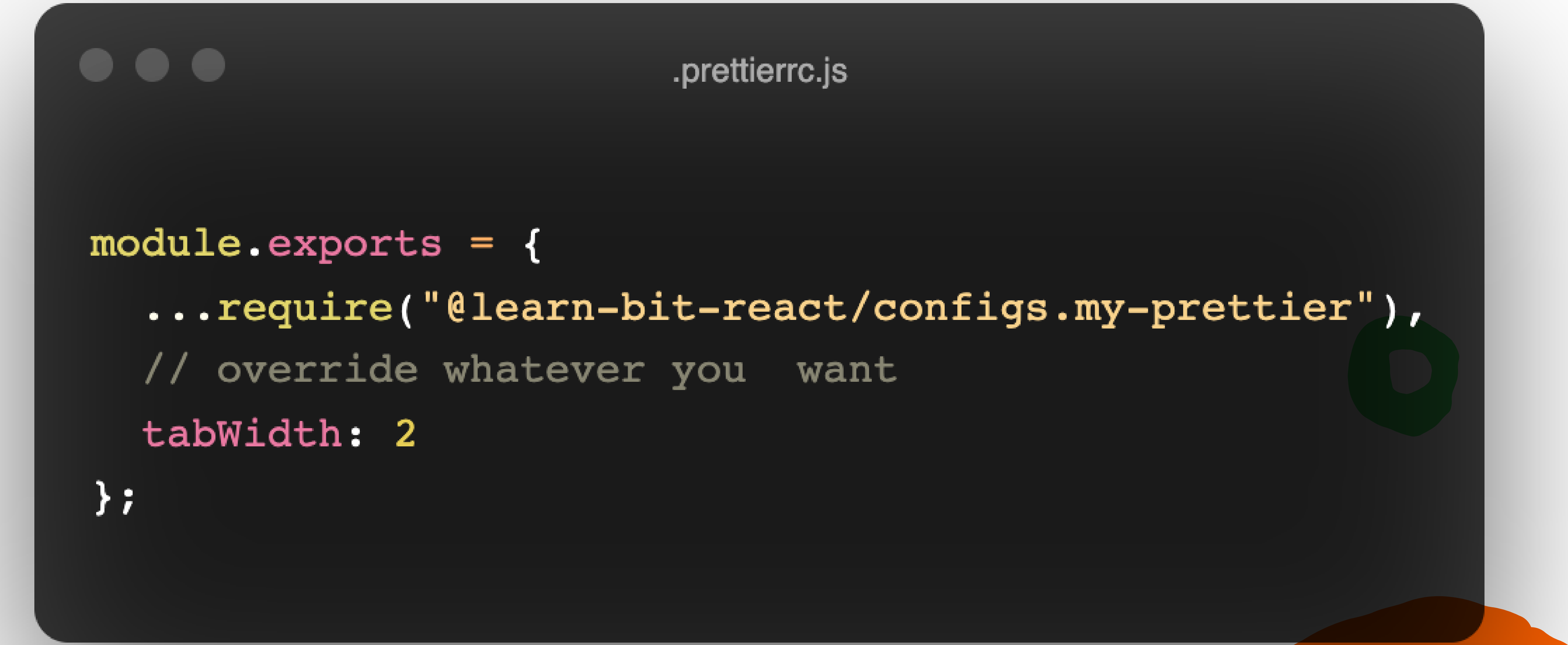


```
module.exports = {
  overrides: [
    {
      files: ['*.ts', '*.tsx', '*.js', '*.jsx', '*.mjs'],
      rules: {
        'no-console': 'error'
        // add a new rule
      }
    }
  ]
};
```

Sharing Prettier config

Create one config
Apply it to all your projects

Make a change
Create a new version



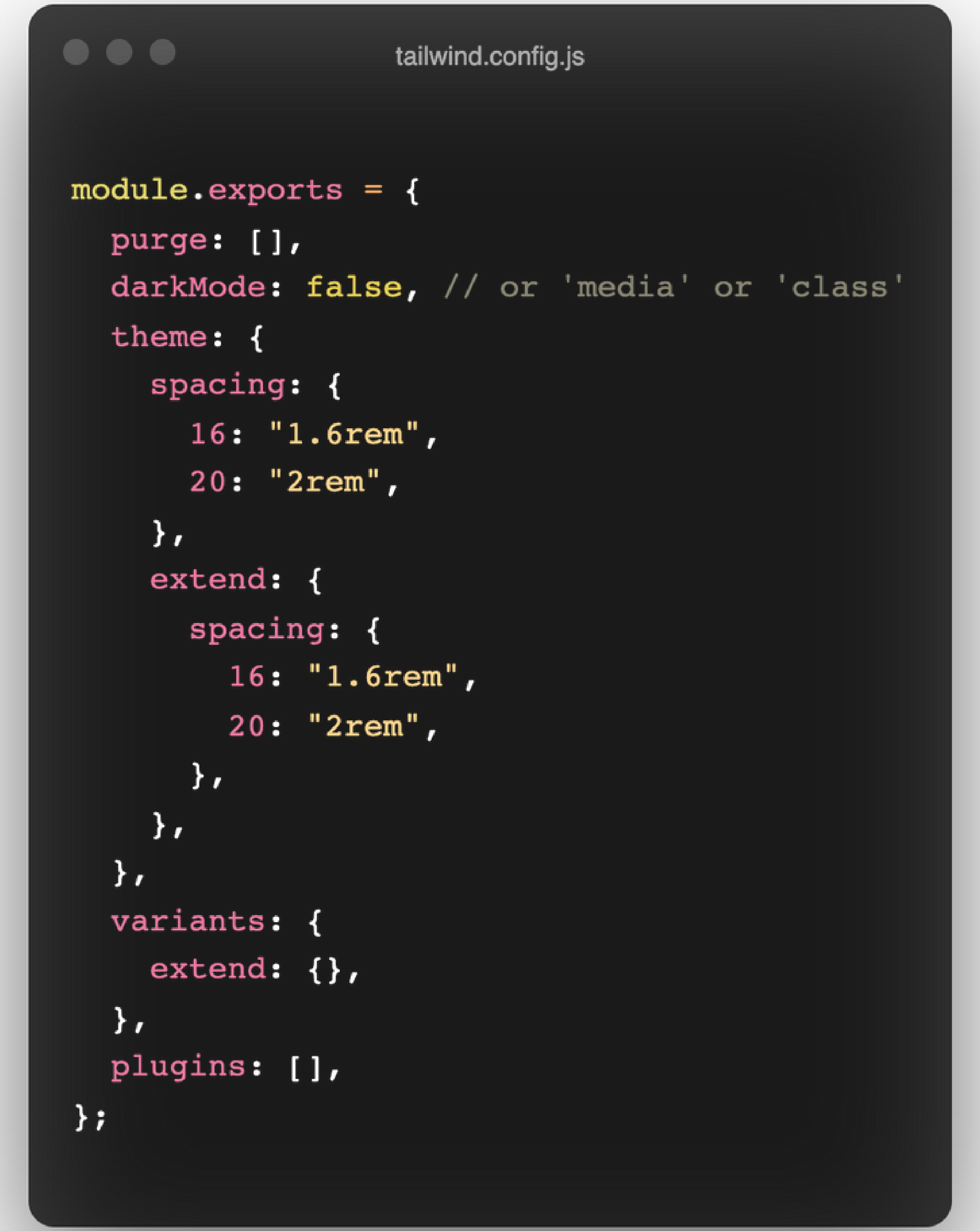
.prettierrc.js

```
module.exports = {
  ...require("@learn-bit-react/configs.my-prettier"),
  // override whatever you  want
  tabWidth: 2
};
```

Sharing Tailwind config

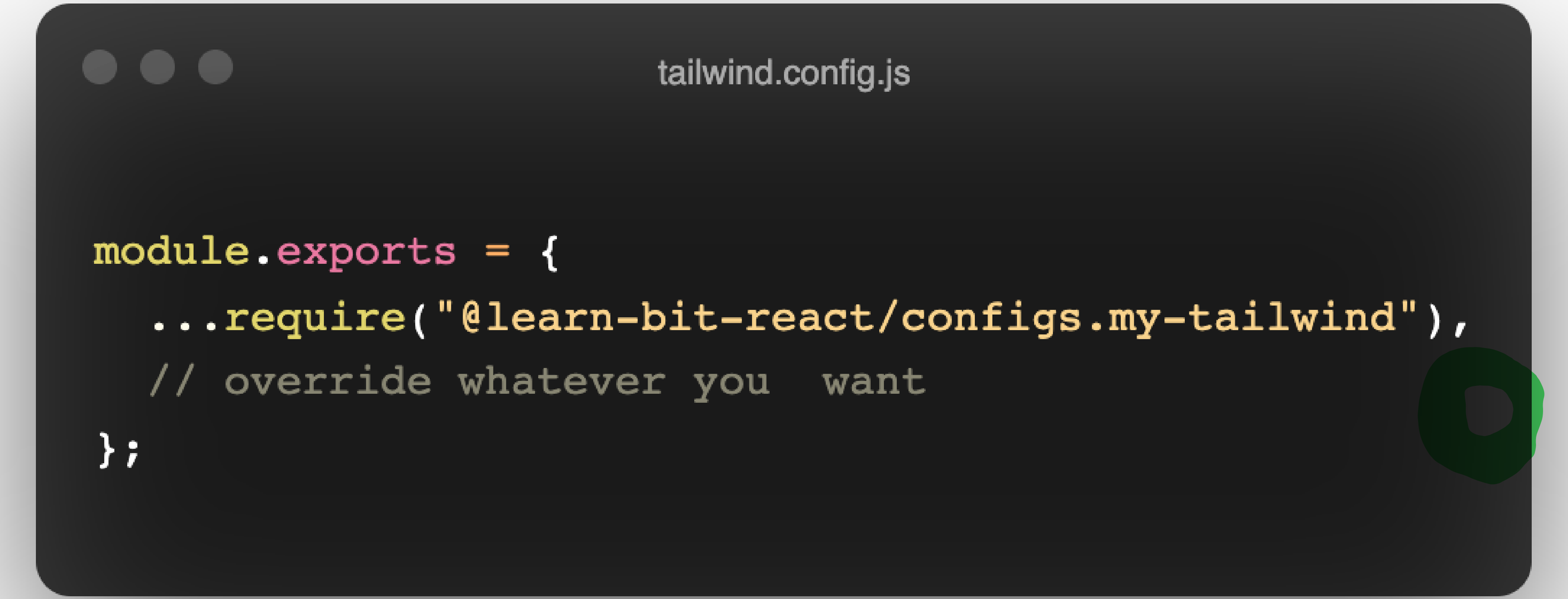
Create one config
Apply it to all your projects

Make a change
Create a new version



```
module.exports = {
  purge: [],
  darkMode: false, // or 'media' or 'class'
  theme: {
    spacing: {
      16: "1.6rem",
      20: "2rem",
    },
    extend: {
      spacing: {
        16: "1.6rem",
        20: "2rem",
      },
    },
  },
  variants: {
    extend: {},
  },
  plugins: [],
};
```

Sharing Tailwind config



tailwind.config.js

```
module.exports = {
  ...require("@learn-bit-react/configs.my-tailwind"),
  // override whatever you  want
};
```

Create one config
Apply it to all your projects

Make a change
Create a new version

Time to play with Generators

Solve the problem once and reuse
Saves time
Saves cognitive load

Time to play with Generators

Solve the problem once and reuse
Saves time
Saves cognitive load

Important!!
Maintain your generators
It's an ongoing project

Thank you

Demo ESLint Component

debs_obrien