no no forget about type annotation im just want to annotations [warn or error] if there is wrong style passed into wrong component imagine like a view component and u passed a fontSize style to it

ChatGPT

2 / 2

I understand your question now. You want to add custom linting rules to catch cases where specific styles are applied to the wrong components in React Native, without necessarily focusing on type annotations. To achieve this, you can use ESLint with custom rules, similar to what was described earlier.

Here's a more focused example of how you can create an ESLint rule to catch instances where the fontSize style is applied to a View component:

Install ESLint and Create a Custom Rule:

If you haven't already, install ESLint and create a custom rule as follows:

bash

Copy code

npm install --save-dev eslint

touch .eslintrc.js

Then, create a custom ESLint rule file for your project, e.g., eslint-rules/disallow-font-size-on-view.js.

Custom ESLint Rule:

In eslint-rules/disallow-font-size-on-view.js, define a custom ESLint rule to disallow the use of fontSize on View components:

javascript

Copy code

module.exports = {

meta: {

type: 'suggestion',

docs: {

description: 'Disallow fontSize style on View components',

},

},

create: function (context) {

return {

JSXAttribute: function (node) {

if (

node.name &&

node.name.type === 'JSXIdentifier' &&

node.name.name === 'fontSize' &&

node.parent &&

node.parent.type === 'JSXOpeningElement' &&

node.parent.name &&

node.parent.name.type === 'JSXIdentifier' &&

node.parent.name.name === 'View'

) {

context.report({

node: node,

message: 'Do not use fontSize on View components.',

});

}

},

};

},

};

Configure ESLint:

In your ESLint configuration (.eslintrc.js), specify the custom rule:

javascript

Copy code

module.exports = {

// ...

plugins: ['react'],

rules: {

// ...

'custom-rules/disallow-font-size-on-view': 'error',

},

};

Run ESLint:

Now, when you run ESLint in your project, it will enforce the custom rule. If you use fontSize on a View component, ESLint will report an error or warning depending on your configuration:

bash

Copy code

npx eslint .

This setup will give you a way to catch instances where specific styles are applied to the wrong components in your React Native codebase. You can create additional custom rules to enforce other style-related constraints as needed.

User

what if i want to export it as module not a single rule how i do it

ChatGPT

If you want to export a collection of ESLint rules as a module rather than a single rule, you can organize your custom rules into a separate npm package/module that can be shared and reused across multiple projects. Here are the steps to create such a module:

Create a New npm Package:

Start by creating a new npm package to house your custom ESLint rules. You can do this by running:

bash

Copy code

mkdir eslint-custom-rules

cd eslint-custom-rules

npm init

Follow the prompts to set up your package. Be sure to provide a proper package name, version, and description.

Install ESLint and Dependencies:

You'll need ESLint and any other dependencies required for your custom rules. Install them in your package:

bash

Copy code

npm install eslint eslint-plugin-react --save-dev

Create a New ESLint Plugin:

Inside your package, create a directory for your ESLint plugin, for example:

bash

Copy code

mkdir eslint-plugin-my-custom-rules

Write Your Custom ESLint Rules:

Inside the eslint-plugin-my-custom-rules directory, you can define your custom rules. Each rule should be in its own file, following ESLint's naming convention (e.g., no-font-size-on-view.js).

Here's an example structure:

vbnet

Copy code

eslint-custom-rules/

├── eslint-plugin-my-custom-rules/

│ ├── no-font-size-on-view.js

│ ├── rule2.js

│ └── ...

├── package.json

└── README.md

Export Your Custom Rules as an ESLint Plugin:

In your eslint-plugin-my-custom-rules directory, create an index.js file to export your custom rules as an ESLint plugin:

javascript

Copy code

module.exports = {

rules: {

'no-font-size-on-view': require('./no-font-size-on-view'),

// Add other custom rules here

},

};

Configure ESLint to Use Your Plugin:

In your ESLint configuration (.eslintrc.js) in your project that will use these custom rules, specify the plugin and rules:

javascript

Copy code

module.exports = {

plugins: ['my-custom-rules'], // Use the name of your plugin

rules: {

'my-custom-rules/no-font-size-on-view': 'error', // Use the plugin name as a prefix

// Configure other custom rules from your plugin

},

};

Publish Your ESLint Plugin:

Publish your custom ESLint plugin to npm by running:

bash

Copy code

npm login # if not logged in

npm publish --access public

Now your custom rules are available as an npm package that can be installed and used in other projects. Others can use your plugin by adding it to their ESLint configurations and specifying the rules from your package.

Remember to follow best practices for creating and documenting ESLint rules and plugins to make them easy to understand and use by others.

User

this code ` module.exports = {

meta: {

type: 'suggestion',

docs: {

description: 'Disallow fontSize style on View components',

},

},

create: function (context) {

return {

JSXAttribute: function (node) {

if (

node.name &&

node.name.type === 'JSXIdentifier' &&

node.name.name === 'fontSize' &&

node.parent &&

node.parent.type === 'JSXOpeningElement' &&

node.parent.name &&

node.parent.name.type === 'JSXIdentifier' &&

node.parent.name.name === 'View'

) {

context.report({

node: node,

message: 'Do not use fontSize on View components.',

});

}

},

};

},

};

` it looks it gives annotation but not fixing the error if found even if u used eslint --fix

ChatGPT

The code you provided for the ESLint custom rule defines a linting rule that can report issues when it detects the use of fontSize on a View component. However, to automatically fix such issues using eslint --fix, you would need to provide a fix function within your custom rule definition. Here's how you can modify your custom rule to include a fix function:

javascript

Copy code

module.exports = {

meta: {

type: 'suggestion',

docs: {

description: 'Disallow fontSize style on View components',

},

fixable: 'code', // Specify that this rule can automatically fix issues

},

create: function (context) {

return {

JSXAttribute: function (node) {

if (

node.name &&

node.name.type === 'JSXIdentifier' &&

node.name.name === 'fontSize' &&

node.parent &&

node.parent.type === 'JSXOpeningElement' &&

node.parent.name &&

node.parent.name.type === 'JSXIdentifier' &&

node.parent.name.name === 'View'

) {

context.report({

node: node,

message: 'Do not use fontSize on View components.',

fix: function (fixer) {

// Provide a fix function to automatically remove the attribute

return fixer.remove(node);

},

});

}

},

};

},

};

Save to grepper

By adding the fix function and specifying 'code' as the fixable value in the meta section of your custom rule, ESLint will attempt to automatically fix the issue when you use the eslint --fix command.

Keep in mind that the fix function should provide the necessary code modifications to correct the issue, in this case, removing the fontSize attribute from the View component. After making this change, you should be able to use eslint --fix to automatically correct instances where the rule detects the use of fontSize on View components.