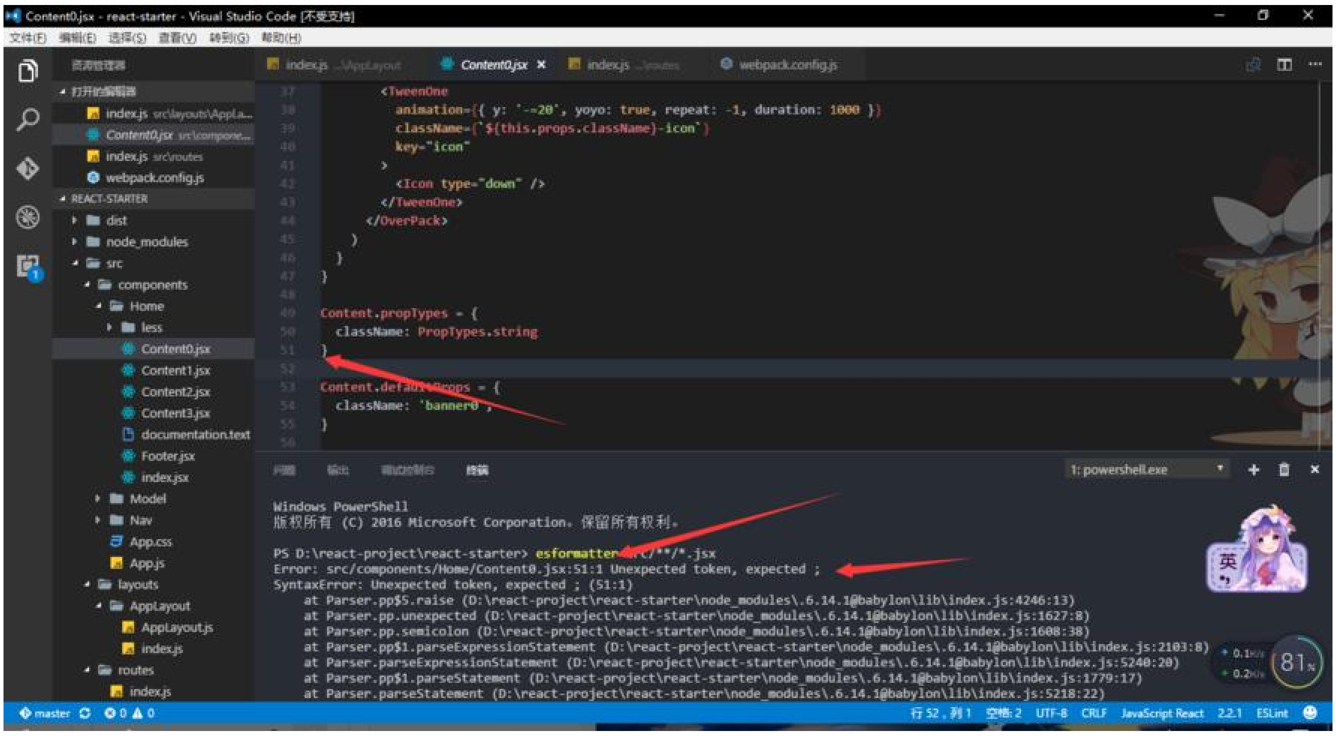
一个莫名其妙的前端问题，求解

一个莫名其妙的报错，求解~

我在配置一个react脚手架的时候，使用了esformatter和eslint。

项目正常运行没有任何错误，但是在使用esformatter命令时，会报如下错误
命令是 esformatter src/**/*.jsx。



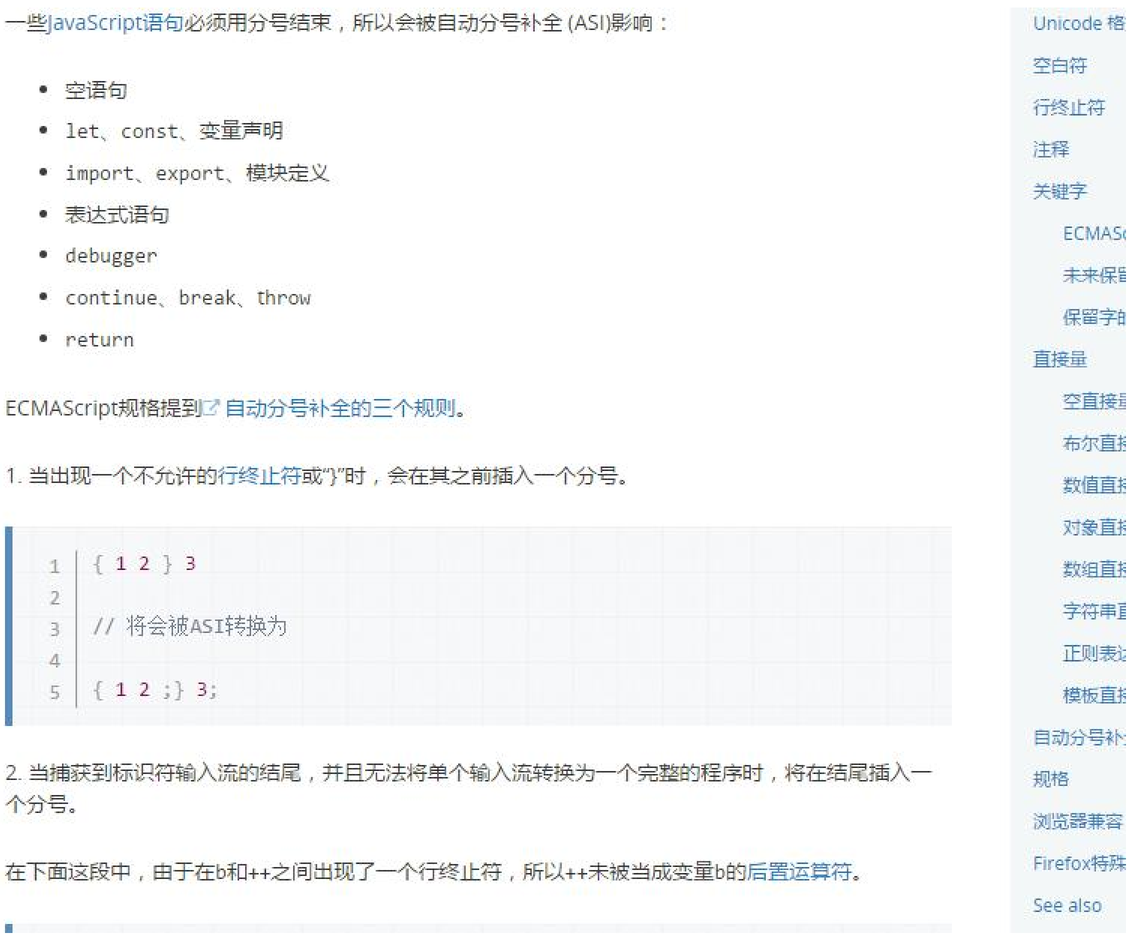
```
37          <TweenOne
38            animation={{ y: '-=20', yoyo: true, repeat: -1, duration: 1000 }}
39            className={`${this.props.className}-icon`}
40            key="icon"
41          >
42            <Icon type="down" />
43          </TweenOne>
44        </OverPack>
45      )
46    }
47  }
48
49  Content.propTypes = {
50    className: PropTypes.string
51  }
52
53  Content.defaultProps = {
54    className: 'banner0',
55  }
56
```

```
Windows PowerShell
版权所有 (C) 2016 Microsoft Corporation。保留所有权利。

PS D:\react-project\react-starter> esformatter src/**/*.jsx
Error: src/components/Home/Content0.jsx:51:1 Unexpected token, expected ;
SyntaxError: Unexpected token, expected ; (51:1)
    at Parser.pp$5.raise (D:\react-project\react-starter\node_modules\.6.14.1@babylon\lib\index.js:4246:13)
    at Parser.pp.unexpected (D:\react-project\react-starter\node_modules\.6.14.1@babylon\lib\index.js:1627:8)
    at Parser.pp.semicolon (D:\react-project\react-starter\node_modules\.6.14.1@babylon\lib\index.js:1608:38)
    at Parser.pp$1.parseExpressionStatement (D:\react-project\react-starter\node_modules\.6.14.1@babylon\lib\index.js:2103:8)
    at Parser.parseExpressionStatement (D:\react-project\react-starter\node_modules\.6.14.1@babylon\lib\index.js:5240:20)
    at Parser.pp$1.parseStatement (D:\react-project\react-starter\node_modules\.6.14.1@babylon\lib\index.js:1779:17)
    at Parser.parseStatement (D:\react-project\react-starter\node_modules\.6.14.1@babylon\lib\index.js:5218:22)
```

运行一切正常(运行时已经进行了eslint进行代码检查，没有问题，eslint将分号去掉的)，但是就是这一个错误..导致一直不能进行格式化~~~，求解啊，就这一个问题了，谢谢各位大神些=-=

经过在网上的一系列搜索大法，貌似、好像、可能、我定位到的是



一些JavaScript语句必须用分号结束，所以会被自动分号补全 (ASI)影响：

- 空语句
- let、const、变量声明
- import、export、模块定义
- 表达式语句
- debugger
- continue、break、throw
- return

ECMAScript规格提到 自动分号补全的三个规则。

1. 当出现一个不允许的行终止符或"}"时，会在其之前插入一个分号。

```
{ 1 2 } 3

// 将会被ASI转换为

{ 1 2 ;} 3;
```

2. 当捕获到标识符输入流的结尾，并且无法将单个输入流转换为一个完整的程序时，将在结尾插入一个分号。

在下面这段中，由于在b和++之间出现了一个行终止符，所以++未被当成变量b的后置运算符。

但是还是不知道咋办QAQ，我的天啦噜，困扰我很久了。

eslint 配置文件如下：

```
{
  "env": {
      "browser": true,
      "commonjs": true,
      "es6": true,
      "node": true
  },
  "extends": ["eslint:recommended","plugin:react/recommended"],
  "parserOptions": {
      "ecmaFeatures": {
          "experimentalObjectRestSpread": true,
          "jsx": true
      },
      "sourceType": "module"
  },
  "plugins": [
      "react"
  ],
  "settings": {
      "react": {
          "createClass": "createClass",
          "pragma": "React",
          "version": "15.4.2"
      }
  },
  "rules": {
      "indent": [
          "error",
          2
      ],//tab占两个空格
      // "linebreak-style": [
      //     "error",
      //     "windows"
      // ], 使用windows结束符
      "quotes": [
          "error",
          "single"
      ],//强制使用单引号
      "semi": [
          "error",
          "never"
      ],
      "no-console":"off"
      // "react/jsx-uses-react": "error",//防止React被错误地标记为未使用
      // "react/jsx-uses-vars": "error",//防止在JSX中使用的变量被错误地标记为未使用
      // "react/no-danger":"error",//防止在jsx中使用危险的属性
      // "react/no-deprecated":"error",//防止使用废弃的方法
      // "react/no-did-mount-set-state":"error",//防止在didmount时setState
      // "react/no-did-update-set-state":"error",//防止在didupdate时setState
      // "react/no-multi-comp":"error",//防止一个文件内定义多个组件
      // "react/prefer-stateless-function":"error",//强制无状态React组件作为纯函数写入
      // "react/react-in-jsx-scope":"error",//使用JSX时防止丢失React
      // "react/require-render-return":"error",//强制ES5或ES6类在render函数中返回值
      // "react/sort-comp":"error",//强制组件方法排序,
      // "react/void-dom-elements-no-children":"error",//闭合标签不能接受子元素
      // "react/jsx-closing-bracket-location":"error",//在JSX中验证关闭括号位置（可修复）
      // "react/jsx-curly-spacing":["error","never"],//禁止在花括号前后使用空格
      // "react/jsx-equals-spacing":["error","always"]//等号前后必须要空格
  }
}
```

---

esformatter 配置文件如下

```
{
  "root": true,
  "plugins": [
  "esformatter-jsx"
],
"jsx": {
  "attrsOnSameLineAsTag": false,
  "maxAttrsOnTag": 5,
  "firstAttributeOnSameLine": true,
  "alignWithFirstAttribute": true,
  "JSXExpressionsSingleLine": true,
  "spaceInJSXExpressionContainers": "",
  "removeSpaceBeforeClosingJSX": true,
  "closingTagOnNewLine": true,
  "JSXAttributeQuotes": "double",
  "formatJSXExpressions":true
},
"indent": {
  "value": "   ",
  "ArrayExpression": 1,
  "AssignmentExpression": 1,
  "BinaryExpression": 1,
  "ConditionalExpression": 1,
  "CallExpression": 1,
  "CatchClause": 1,
  "DoWhileStatement": 1,
  "ForInStatement": 1,
  "ForStatement": 1,
  "FunctionDeclaration": 1,
  "FunctionExpression": 1,
    "IfStatement": 1,
    "MemberExpression": 1,
    "MultipleVariableDeclaration": 1,
    "ObjectExpression": 1,
    "ReturnStatement": ">=1",
    "SwitchCase": 1,
    "SwitchStatement": 1,
    "TopLevelFunctionBlock": 1,
    "TryStatement": 1,
    "VariableDeclaration.LogicalExpression": 1,
    "WhileStatement": 1
  },
  "lineBreak": {
    "before": {
      "AssignmentExpression": ">=0",
      "AssignmentOperator": 0,
      "BlockStatement": 1,
      "CallExpression": -1,
      "ConditionalExpression": ">=1",
      "CatchOpeningBrace": 0,
      "CatchClosingBrace": ">=1",
      "CatchKeyword": 0,
      "DeleteOperator": ">=1",
      "DoWhileStatement": ">=1",
      "DoWhileStatementOpeningBrace": 0,
      "DoWhileStatementClosingBrace": ">=1",
      "EndOfFile": 1,
      "EmptyStatement": -1,
      "FinallyKeyword": -1,
      "FinallyOpeningBrace": 0,
      "FinallyClosingBrace": ">=1",
      "ForInStatement": ">=1",
      "ForInStatementExpressionOpening": 0,
      "ForInStatementExpressionClosing": 0,
      "ForInStatementOpeningBrace": 0,
      "ForInStatementClosingBrace": ">=1",
      "ForStatement": ">=1",
      "ForStatementExpressionOpening": 0,
      "ForStatementExpressionClosing": "<2",
      "ForStatementOpeningBrace": 0,
      "ForStatementClosingBrace": ">=1",
      "FunctionExpression": 0,
      "FunctionExpressionOpeningBrace": 0,
      "FunctionExpressionClosingBrace": ">=1",
      "FunctionDeclaration": ">=1",
      "FunctionDeclarationOpeningBrace": 0,
      "FunctionDeclarationClosingBrace": ">=1",
      "IfStatement": ">=1",
      "IfStatementOpeningBrace": 0,
      "IfStatementClosingBrace": ">=1",
      "ElseIfStatement": 0,
      "ElseIfStatementOpeningBrace": 0,
      "ElseIfStatementClosingBrace": ">=1",
      "ElseStatement": 0,
          "ElseStatementOpeningBrace": 0,
          "ElseStatementClosingBrace": ">=1",
          "LogicalExpression": -1,
          "ObjectExpressionOpeningBrace": 0,
          "ObjectExpressionClosingBrace": 1,
          "Property": 1,
          "ReturnStatement": ">=1",
          "SwitchOpeningBrace": 0,
          "SwitchClosingBrace": ">=1",
          "ThisExpression": -1,
          "ThrowStatement": ">=1",
          "TryKeyword": -1,
          "TryOpeningBrace": 0,
          "TryClosingBrace": ">=1",
          "VariableName": ">=1",
          "VariableValue": 0,
          "VariableDeclaration": ">=1",
          "VariableDeclarationWithoutInit": ">=1",
          "WhileStatement": ">=1",
          "WhileStatementOpeningBrace": 0,
          "WhileStatementClosingBrace": ">=1",
          "ArrayExpressionClosing": 1
        },
        "after": {
          "AssignmentExpression": ">=1",
          "AssignmentOperator": 0,
          "BlockStatement": 0,
      "CallExpression": -1,
      "CatchOpeningBrace": ">=1",
      "CatchClosingBrace": ">=0",
      "CatchKeyword": 0,
      "ConditionalExpression": ">=1",
      "DeleteOperator": ">=1",
      "DoWhileStatement": ">=1",
      "DoWhileStatementOpeningBrace": ">=1",
      "DoWhileStatementClosingBrace": 0,
      "EmptyStatement": -1,
      "FinallyKeyword": -1,
      "FinallyOpeningBrace": ">=1",
      "FinallyClosingBrace": ">=1",
      "ForInStatement": ">=1",
      "ForInStatementExpressionOpening": "<2",
      "ForInStatementExpressionClosing": -1,
      "ForInStatementOpeningBrace": ">=1",
      "ForInStatementClosingBrace": ">=1",
      "ForStatement": ">=1",
      "ForStatementExpressionOpening": "<2",
      "ForStatementExpressionClosing": -1,
      "ForStatementOpeningBrace": ">=1",
      "ForStatementClosingBrace": ">=1",
      "FunctionExpression": ">=1",
      "FunctionExpressionOpeningBrace": ">=1",
      "FunctionExpressionClosingBrace": -1,
      "FunctionDeclaration": ">=1",
      "FunctionDeclarationOpeningBrace": ">=1",

      "FunctionDeclarationClosingBrace": ">=1",

      "IfStatement": ">=1",

      "IfStatementOpeningBrace": ">=1",

      "IfStatementClosingBrace": ">=1",

      "ElseIfStatement": ">=1",

      "ElseIfStatementOpeningBrace": ">=1",

      "ElseIfStatementClosingBrace": ">=1",

      "ElseStatement": ">=1",

      "ElseStatementOpeningBrace": ">=1",

      "ElseStatementClosingBrace": ">=1",

      "LogicalExpression": -1,

      "ObjectExpressionClosingBrace": 0,

      "ObjectExpressionOpeningBrace": 1,

      "PropertyValue": 0,

      "Property": 0,

      "ReturnStatement": 1,

      "SwitchOpeningBrace": ">=1",

      "SwitchClosingBrace": ">=1",

      "ThisExpression": 0,

      "ThrowStatement": ">=1",

      "TryKeyword": -1,

      "TryOpeningBrace": ">=1",

      "TryClosingBrace": 0,

      "VariableDeclaration": ">=1",

      "WhileStatement": ">=1",

      "WhileStatementOpeningBrace": ">=1",
            "WhileStatementClosingBrace": ">=1",
            "ArrayExpressionOpening": 1,
            "ArrayExpressionComma": 1
          }
        },
        "whiteSpace": {
          "value": " ",
          "removeTrailing": 1,
          "before": {
            "ArgumentComma": 0,
            "ArgumentList": 1,
            "ArgumentListArrayExpression": 0,
            "ArgumentListFunctionExpression": 1,
            "ArgumentListObjectExpression": 0,
            "AssignmentOperator": 1,
            "BinaryExpression": 0,
            "BinaryExpressionOperator": 1,
            "BlockComment": 1,
            "CallExpression": 1,
            "CatchParameterList": 0,
            "CatchOpeningBrace": 1,
            "CatchClosingBrace": 1,
            "CatchKeyword": 1,
            "CommaOperator": 0,
            "ConditionalExpressionConsequent": 1,
            "ConditionalExpressionAlternate": 1,
            "DoWhileStatementOpeningBrace": 1,
      "DoWhileStatementClosingBrace": 1,

      "DoWhileStatementConditional": 1,

      "EmptyStatement": 0,

      "ExpressionClosingParentheses": 0,

      "FinallyKeyword": -1,

      "FinallyOpeningBrace": 1,

      "FinallyClosingBrace": 1,

      "ForInStatement": 1,

      "ForInStatementExpressionOpening": 1,

      "ForInStatementExpressionClosing": 0,

      "ForInStatementOpeningBrace": 1,

      "ForInStatementClosingBrace": 1,

      "ForStatement": 1,

      "ForStatementExpressionOpening": 1,

      "ForStatementExpressionClosing": 0,

      "ForStatementOpeningBrace": 1,

      "ForStatementClosingBrace": 1,

      "ForStatementSemicolon": 0,

      "FunctionDeclarationOpeningBrace": 1,

      "FunctionDeclarationClosingBrace": 1,

      "FunctionExpressionOpeningBrace": 1,

      "FunctionExpressionClosingBrace": 1,

      "IfStatementConditionalOpening": 1,

      "IfStatementConditionalClosing": 1,

      "IfStatementOpeningBrace": 1,

      "IfStatementClosingBrace": 1,

      "ElseStatementOpeningBrace": 1,
      "ElseStatementClosingBrace": 1,

      "ElseIfStatementOpeningBrace": 1,

      "ElseIfStatementClosingBrace": 1,

      "MemberExpressionClosing": 1,

      "LineComment": 1,

      "LogicalExpressionOperator": 1,

      "PropertyName": 1,

      "Property": 1,

      "PropertyValue": 1,

      "ParameterList": 1,

      "SwitchDiscriminantOpening": 1,

      "SwitchDiscriminantClosing": 0,

      "ThrowKeyword": 1,

      "TryKeyword": -1,

      "TryOpeningBrace": 1,

      "TryClosingBrace": 1,

      "UnaryExpressionOperator": 0,

      "VariableName": 1,

      "VariableValue": 1,

      "WhileStatementConditionalOpening": 1,

      "WhileStatementConditionalClosing": 0,

      "WhileStatementOpeningBrace": 1,

      "WhileStatementClosingBrace": 1,

      "ParameterComma": 0,

      "ArrayExpressionOpening": 1,

      "ArrayExpressionClosing": 1,

      "ArrayExpressionComma": 0,
        "ObjectExpressionClosingBrace": 1
      },
      "after": {
        "ObjectExpressionOpeningBrace": 1,
        "ArrayExpressionOpening": 1,
        "ArrayExpressionClosing": 0,
        "ArrayExpressionComma": 1,
        "ArgumentComma": 1,
        "ArgumentList": 1,
        "ArgumentListArrayExpression": 1,
        "ArgumentListFunctionExpression": 1,
        "ArgumentListObjectExpression": 0,
        "AssignmentOperator": 1,
        "BinaryExpression": 0,
        "BinaryExpressionOperator": 1,
        "BlockComment": 1,
        "CallExpression": 0,
        "CatchParameterList": 0,
        "CatchOpeningBrace": 1,
        "CatchClosingBrace": 1,
        "CatchKeyword": 1,
        "CommaOperator": 1,
        "ConditionalExpressionConsequent": 1,
        "ConditionalExpressionTest": 1,
        "DoWhileStatementOpeningBrace": 1,
        "DoWhileStatementClosingBrace": 1,
        "DoWhileStatementBody": 1,
      "EmptyStatement": 0,
      "ExpressionOpeningParentheses": 0,
      "FinallyKeyword": -1,
      "FinallyOpeningBrace": 1,
      "FinallyClosingBrace": 1,
      "ForInStatement": 1,
      "ForInStatementExpressionOpening": 0,
      "ForInStatementExpressionClosing": 1,
      "ForInStatementOpeningBrace": 1,
      "ForInStatementClosingBrace": 1,
      "ForStatement": 1,
      "ForStatementExpressionOpening": 0,
      "ForStatementExpressionClosing": 1,
      "ForStatementClosingBrace": 1,
      "ForStatementOpeningBrace": 1,
      "ForStatementSemicolon": 0,
      "FunctionReservedWord": 1,
      "FunctionName": 0,
      "FunctionExpressionOpeningBrace": 0,
      "FunctionExpressionClosingBrace": 0,
      "FunctionDeclarationOpeningBrace": 1,
      "FunctionDeclarationClosingBrace": 1,
      "IfStatementConditionalOpening": 1,
      "IfStatementConditionalClosing": 1,
      "IfStatementOpeningBrace": 1,
      "IfStatementClosingBrace": 1,
      "ElseStatementOpeningBrace": 1,
          }

              "ElseStatementClosingBrace": 1,

              "ElseIfStatementOpeningBrace": 1,

              "ElseIfStatementClosingBrace": 1,

              "MemberExpressionOpening": 1,

              "LogicalExpressionOperator": 1,

              "PropertyName": 0,

              "PropertyValue": 0,

              "ParameterComma": 1,

              "ParameterList": 1,

              "SwitchDiscriminantOpening": 0,

              "SwitchDiscriminantClosing": 1,

              "ThrowKeyword": 1,

              "TryKeyword": -1,

              "TryOpeningBrace": 1,

              "TryClosingBrace": 1,

              "UnaryExpressionOperator": 0,

              "VariableName": 1,

              "WhileStatementConditionalOpening": 0,

              "WhileStatementConditionalClosing": 1,

              "WhileStatementOpeningBrace": 1,

              "WhileStatementClosingBrace": 1,

              "ObjectExpressionClosingBrace": 0

            }

          }

        }
```