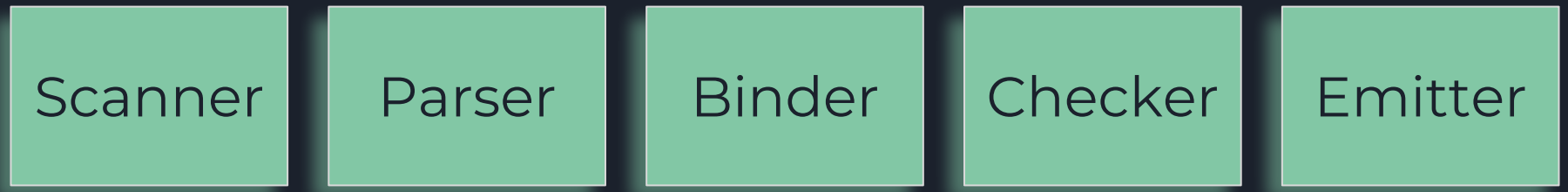
Writing custom TSLint rules

And learning about TypeScript in the process

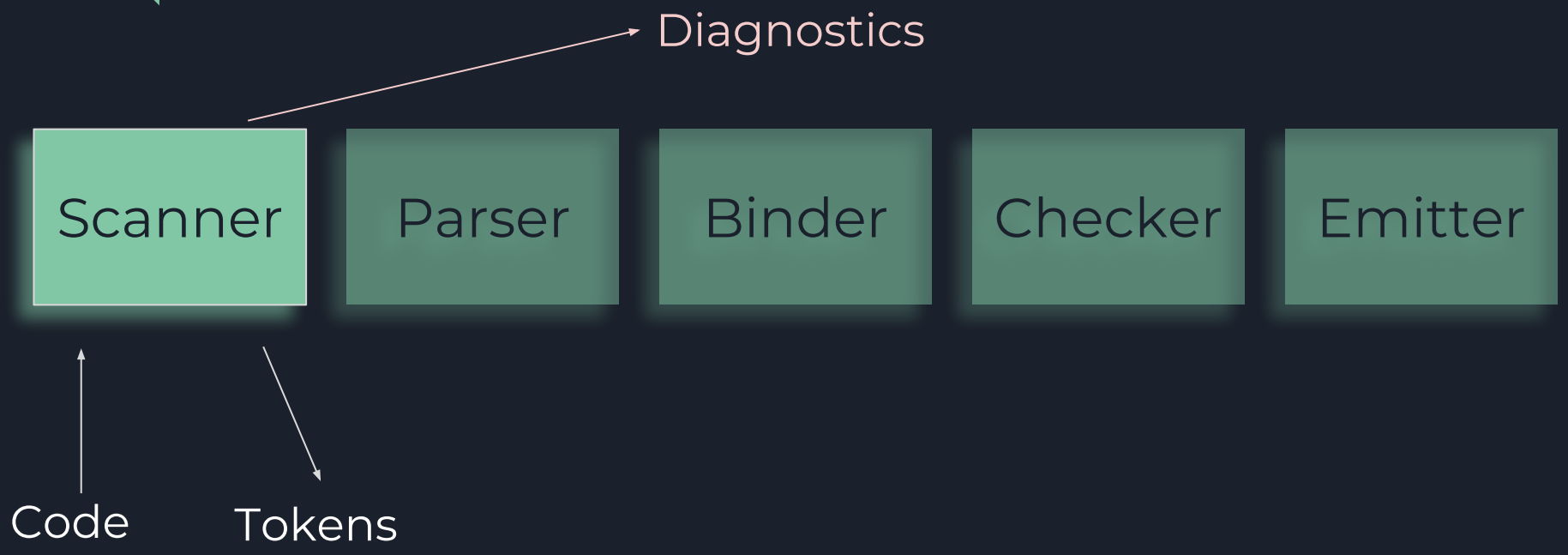
What does TypeScript do?

TypeScript Compiler Architecture



TypeScript Compiler Architecture

TypeScript Compiler Architecture



Scanner - token stream

```
let foo = 5;
foo **= 2;
```

Input: Code

```
LetKeyword
Identifier
EqualsToken
NumericLiteral
SemicolonToken
Identifier
AsteriskAsteriskEqualsToken
NumericLiteral
SemicolonToken
EndOfFileToken
```

Output: Tokens

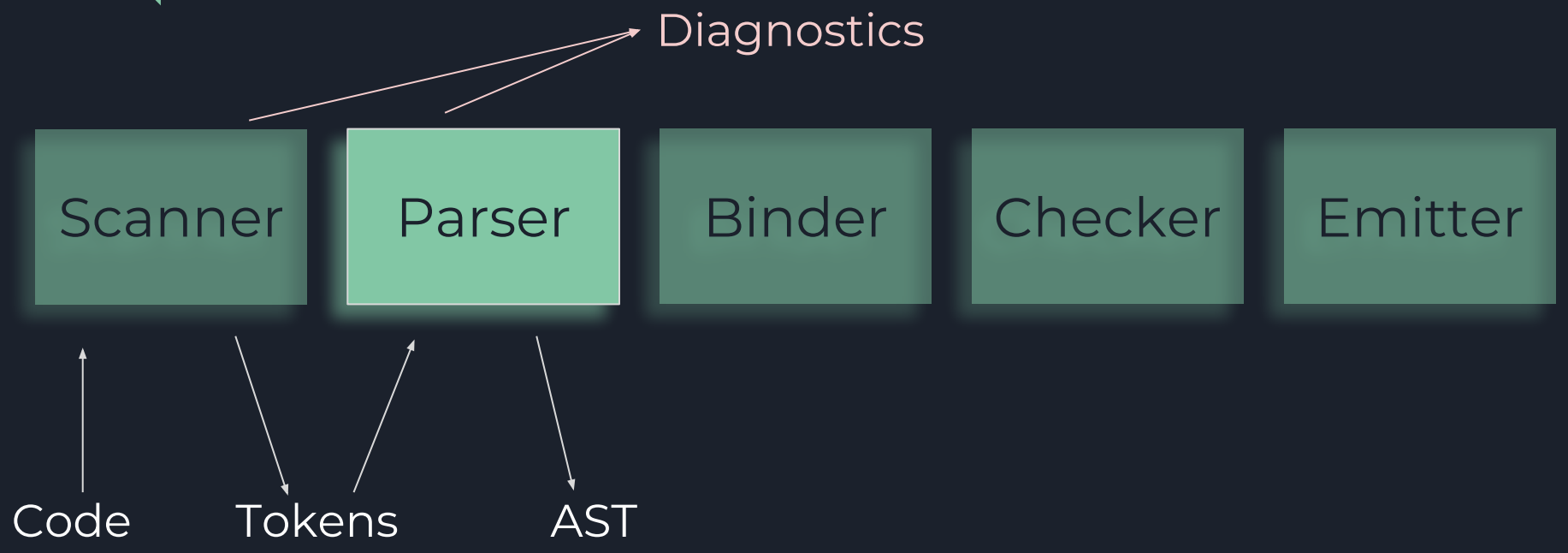
Scanner - diagnostics

Example hex number: `0xFF6767`

```
if (pos + 2 < end && (text.charCodeAt(pos + 1) === CharacterCodes.X
    || text.charCodeAt(pos + 1) === CharacterCodes.x)
) {
    pos += 2;
    let value = scanMinimumNumberOfHexDigits(1, /*canHaveSeparators*/ true);
    if (value < 0) {
        error(Diagnostics.Hexadecimal_digit_expected);
        value = 0;
    }
    tokenValue = "" + value;
    tokenFlags |= TokenFlags.HexSpecifier;
    return token = SyntaxKind.NumericLiteral;
}
```

TypeScript Compiler Architecture

TypeScript Compiler Architecture



Parser

```
LetKeyword
Identifier
EqualsToken
NumericLiteral
SemicolonToken
Identifier
AsteriskAsteriskEqualsToken
NumericLiteral
SemicolonToken
EndOfFileToken
```

Input: Tokens

- SourceFile
  - VariableStatement
    - VariableDeclarationList
      - VariableDeclaration
        - Identifier
        - NumericLiteral
  - ExpressionStatement
    - BinaryExpression
      - Identifier
      - AsteriskAsteriskEqualsToken
      - NumericLiteral
  - EndOfFileToken

Output: AST

Parser - Abstract Syntax Tree (AST)

```
let foo = 5;
foo **= 2;
```

Code

- SourceFile
  - VariableStatement
    - VariableDeclarationList
      - VariableDeclaration
        - Identifier
        - NumericLiteral
  - ExpressionStatement
    - BinaryExpression
      - Identifier
      - AsteriskAsteriskEqualsToken
      - NumericLiteral
  - EndOfFileToken

AST

Parser - AST

```
let foo = 5;
foo **= 2;
```

Code

```
▾ SourceFile
  ▾ VariableStatement
    ▾ VariableDeclarationList
      ▾ VariableDeclaration
          Identifier
          NumericLiteral
  ▾ ExpressionStatement
    ▾ BinaryExpression
        Identifier
        AsteriskAsteriskEqualsToken
        NumericLiteral
  EndOfFileToken
```

AST

```
▾ BinaryExpression
  pos: 12
  end: 22
  flags: 0
  transformFlags: 536870944
  kind: 199 (SyntaxKind.BinaryExpression)
  left: {
    ▸ Identifier
  }
  operatorToken: {
    ▸ AsteriskAsteriskEqualsToken
  }
  right: {
    ▸ NumericLiteral
  }
  id: 5
  getChildCount(): 3
  getFullStart(): 12
  getStart(): 13
  getStart(sourceFile, true): 13
  getFullWidth(): 10
  getWidth(): 9
  getLeadingTriviaWidth(): 1
  getFullText(): "\nfoo **= 2"
  getText(): "foo **= 2"
```

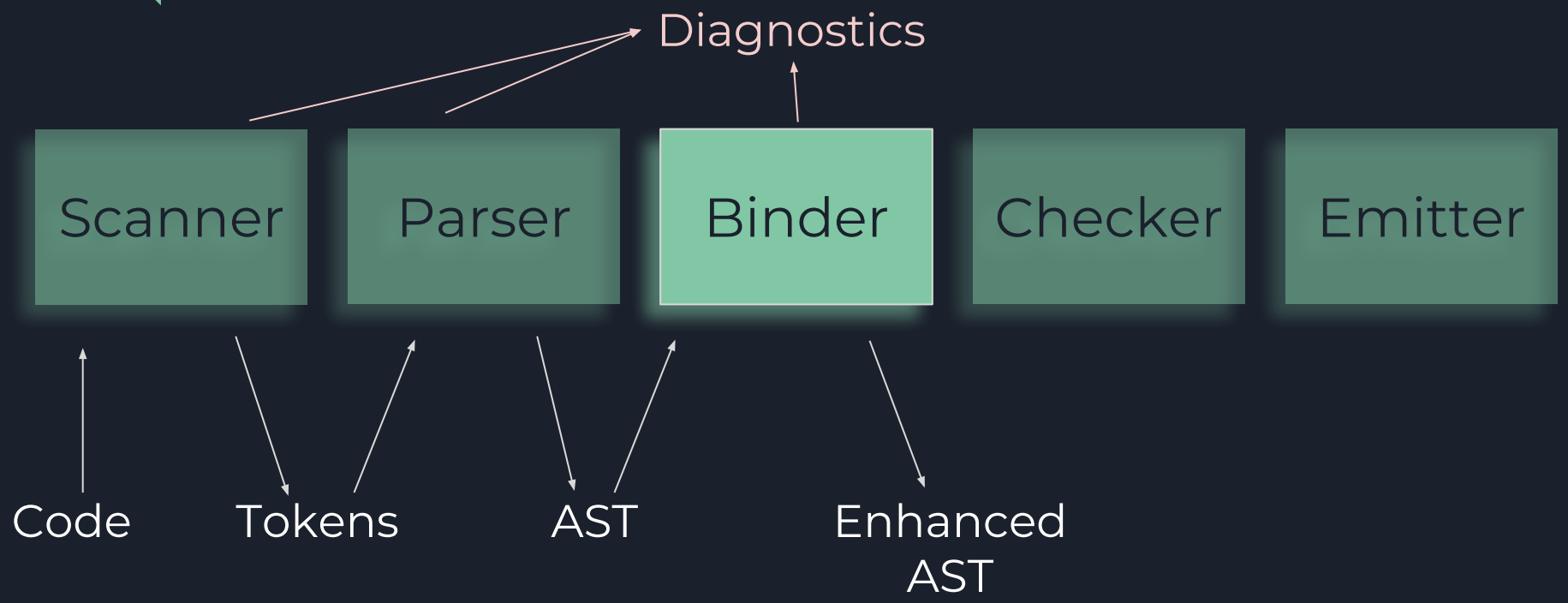
AST Node

Parser - diagnostics

```
function parseJsxChild(openingTag: JsxOpeningElement | JsxOpeningFragment, toker
    switch (token) {
        case SyntaxKind.EndOfFileToken:
            // If we hit EOF, issue the error at the tag that lacks the closing
            // rather than at the end of the file (which is useless)
            if (isJsxOpeningFragment(openingTag)) {
                parseErrorAtRange(
                    openingTag,
                    Diagnostics.JSX_fragment_has_no_corresponding_closing_tag,
                );
            }
```

TypeScript Compiler Architecture

TypeScript Compiler Architecture



Binder - generates symbol tables

```
let foo = 5;
foo **= 2;
```

```
const symbol = {
    flags: SymbolFlags.BlockScopedVariable,
    escapedName: "foo",
    declarations: {}, // some AST Node
    valueDeclaration: {}, // some AST Node
}
```

Binder - computes transformation info

```
function computeCatchClause(node: CatchClause, subtreeFlags: TransformFlags) {
    let transformFlags = subtreeFlags;

    if (!node.variableDeclaration) {
        transformFlags |= TransformFlags.AssertESNext;
    }
    else if (isBindingPattern(node.variableDeclaration.name)) {
        transformFlags |= TransformFlags.AssertES2015;
    }

    node.transformFlags = transformFlags | TransformFlags.HasComputedFlags;
    return transformFlags & ~TransformFlags.CatchClauseExcludes;
}
```

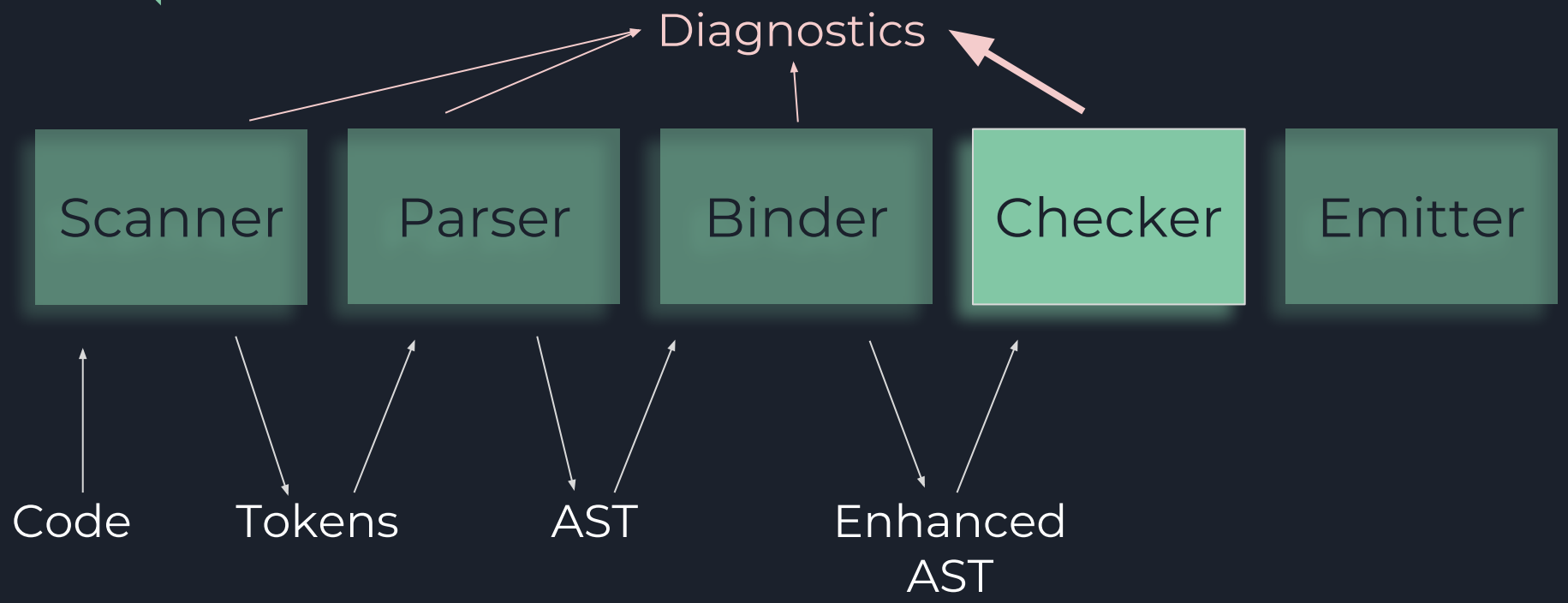
Binder - diagnostics

```
function checkStrictModeNumericLiteral(node: NumericLiteral) {
    if (inStrictMode && node.numericLiteralFlags & TokenFlags.Octal) {
        file.bindDiagnostics.push(
            createDiagnosticForNode(node, Diagnostics.Octal_literals_are_not_allowed_in_strict_mode),
        );
    }
}
```

Note: In strict mode, octal literals such as ‘`0777`’ aren’t allowed.
However, there’s a newer syntax, ‘`0o777`’, which is fine to use.

TypeScript Compiler Architecture

TypeScript Compiler Architecture



Checker - diagnostics

Branch: master ▾ TypeScript / src / compiler / checker.ts Find file Copy path

ahejlsberg Merge pull request #25817 from Microsoft/fixGenericRestTypes 7c512fb 2 days ago

126 contributors and others

1.65 MB Download History

View Raw

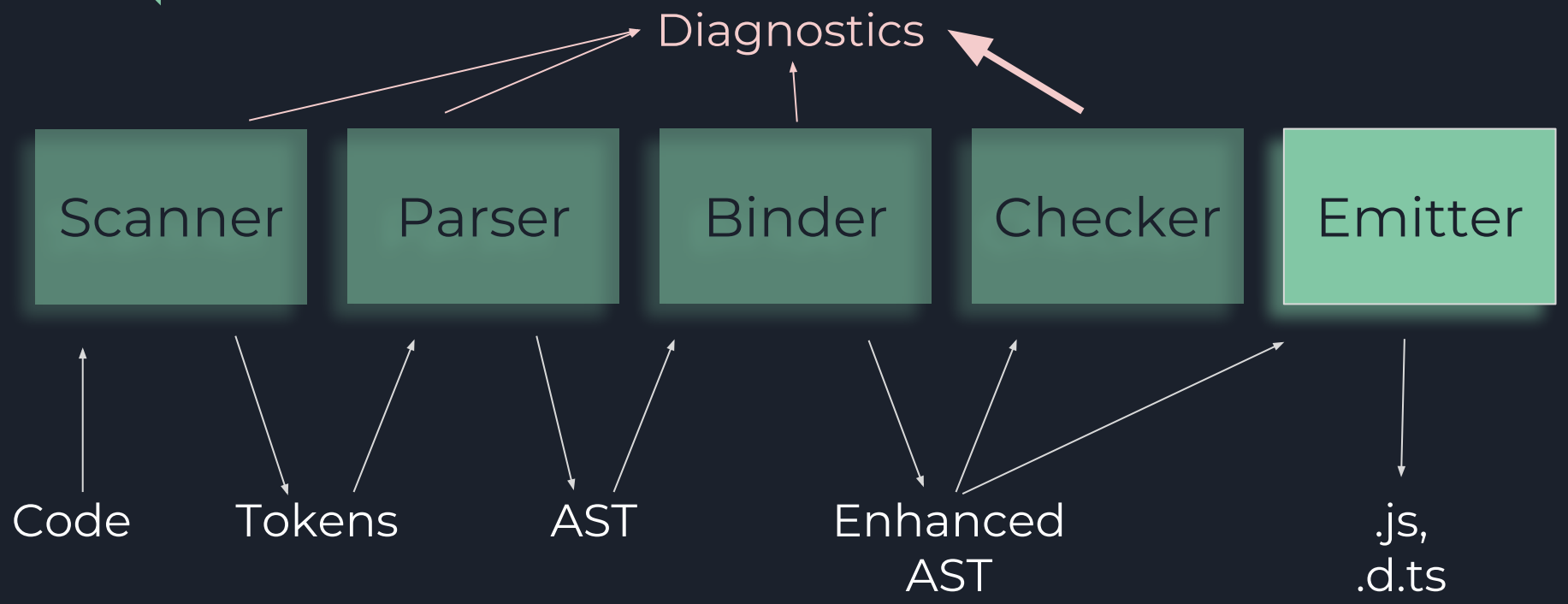
(Sorry about that, but we can't show files that are this big right now.)

30,000 lines of TypeScripty goodness

Checker - diagnostics

```
if (isReadonlySymbol(localOrExportSymbol)) {
    error(
        node,
        Diagnostics.Cannot_assign_to_0_because_it_is_a_constant_or_a_read_only_property,
        symbolToString(symbol),
    );
    return errorType;
}
```

TypeScript Compiler Architecture



TSLint - writing your own rules

Live coding

Does your rule require type info?

If no, extend **`AbstractRule`** and implement:

```
abstract apply(sourceFile: ts.SourceFile): RuleFailure[];
```

If yes, extend **`TypedRule`** and implement:

```
abstract applyWithProgram(sourceFile: ts.SourceFile, program: ts.Program): RuleFailure[];
```

If optionally, extend **`OptionallyTypedRule`** and implement both of the above.

Autofixing is great!

TypeScript Resources

Tools:

- TypeScript AST Viewer
- AST Explorer

TypeScript Compiler Documentation *(warning, not all resources are necessarily up to date / comprehensive)*:

- TypeScript Deep Dive - Compiler Internals (contains TS API usage examples)
- Typescript Architectural Overview

TSLint Developer Resources

Developer documentation:

- Developing custom rules
- Testing custom rules
- Performance tips

Examples:

- Very simple custom rule example (from demo)
- Custom rules from all over