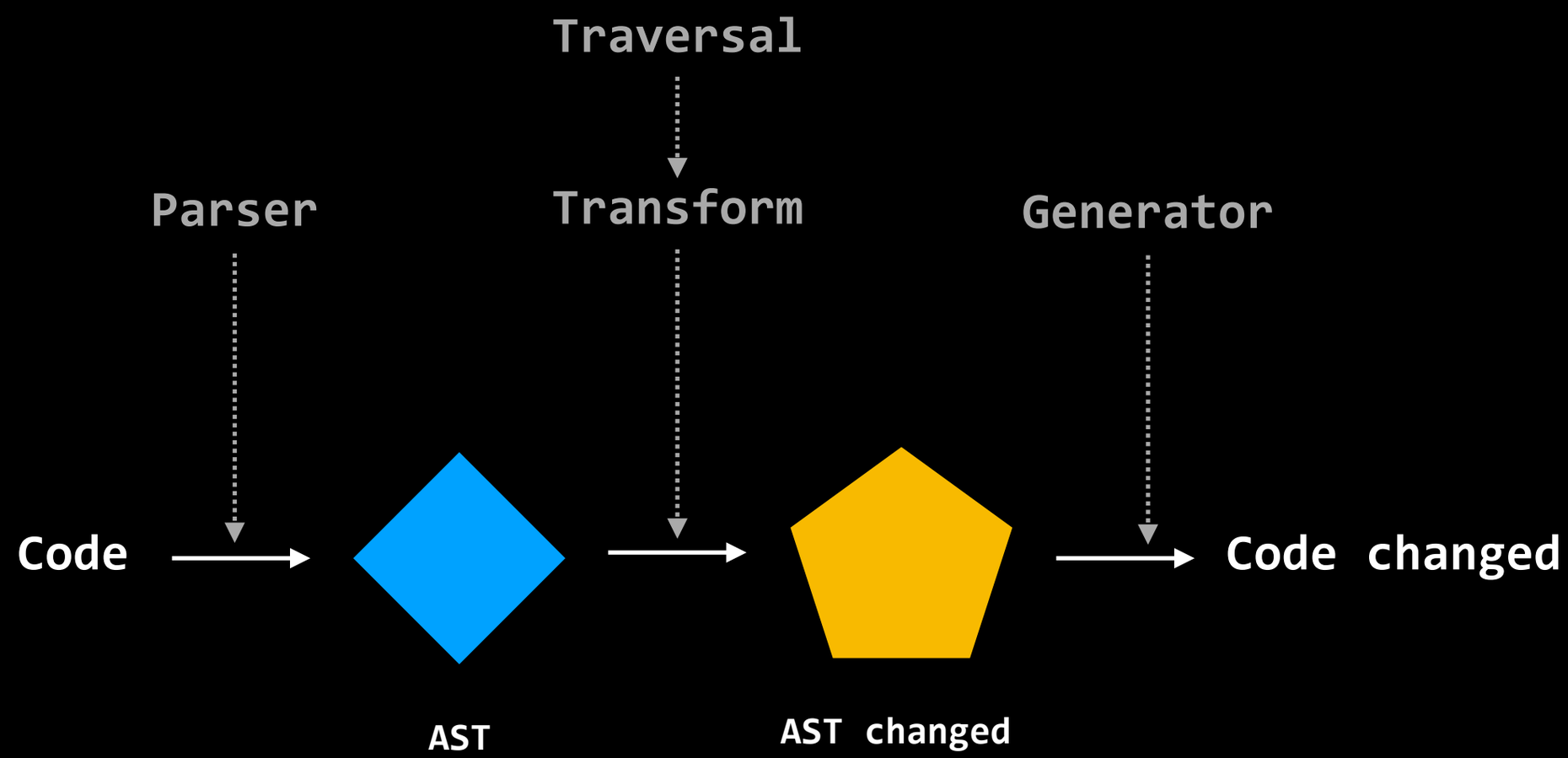
使用 babel 进行 AST 分析和处理

@小胡子哥

Flow


Traversal
Parser
Transform
Generator
Code
AST
AST changed
Code changed

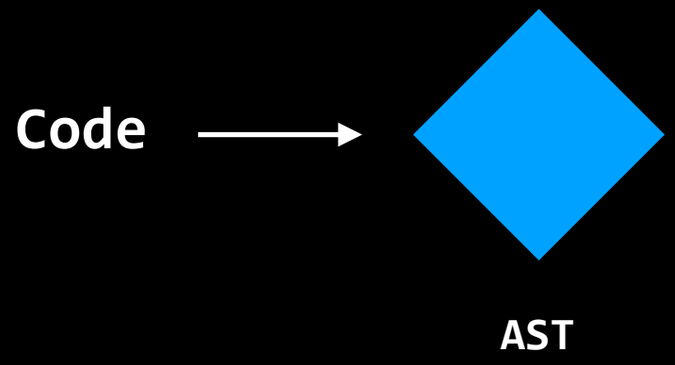
AST

```
function plus(a, b) {
  return a + b;
}
```

```
interface Function <: Node {
  id: Identifier | null;
  params: [ Pattern ];
  body: BlockStatement;
  generator: boolean;
  async: boolean;
}
```

https://astexplorer.net/

Parser




```
const babylon = require('babylon');

const code = `
function plus(a, b) {
  return a + b;
}
`;

const ast = babylon.parse(code, {
  sourceType: 'module'
});
```

Traversal

```
function plus(a, b) {
  return a + b;
}
```

```
- Program
  - FunctionDeclaration
    - Identifier plus
    - Identifier plus
    - Identifier a
    - Identifier a
    - Identifier b
    - Identifier b
    - BlockStatement
      - ReturnStatement
        - BinaryExpression
          - Identifier a
          - Identifier a
          - Identifier b
          - Identifier b
        - BinaryExpression
      - ReturnStatement
    - BlockStatement
  - FunctionDeclaration
- Program
```

Traversal

```
const traversal = require('babel-traverse').default;

traversal(ast, {
  Identifier: {
    enter: function (path) {
      console.log(path.node.name, 'enter');
    },
    exit: function (path) {
      console.log(path.node.name, 'exit\n');
    }
  }
});
```

Traversal

```
const code = `
function plus(a, b) {
  return a + b;
}
function minus(a, b) {
  return a - b;
}
`;

// 局部遍历
traversal(ast, {
  FunctionDeclaration: function (path) {
    if (path.node.id.name !== 'plus') return;
    path.traverse({
      Identifier: {
        enter: function (path) {
          console.log(path.node.name, 'enter');
        },
        exit: function (path) {
          console.log(path.node.name, 'exit\n');
        }
      }
    });
  }
});
```

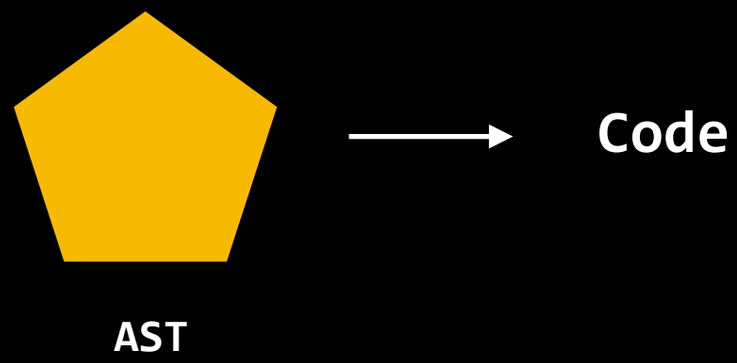
Transform

```
const types = require('babel-types');

traversal(ast, {
  FunctionDeclaration: function (path) {
    path.traverse({
      Identifier: {
        enter: function (path) {
          if (types.isIdentifier(path.node, { name: "a" })) {
            // 节点替换
            path.replaceWith(types.Identifier('x'), path.node);
          }
        },
        exit: function (path) {
          console.log(path.node.name);
        }
      }
    });
  }
});

// function plus(x, b) {
//   return x + b;
// }
```

Generator




```
const generator = require('babel-generator').default;

const changedCode = generator(ast).code;
console.log(changedCode);
```

Tools - template

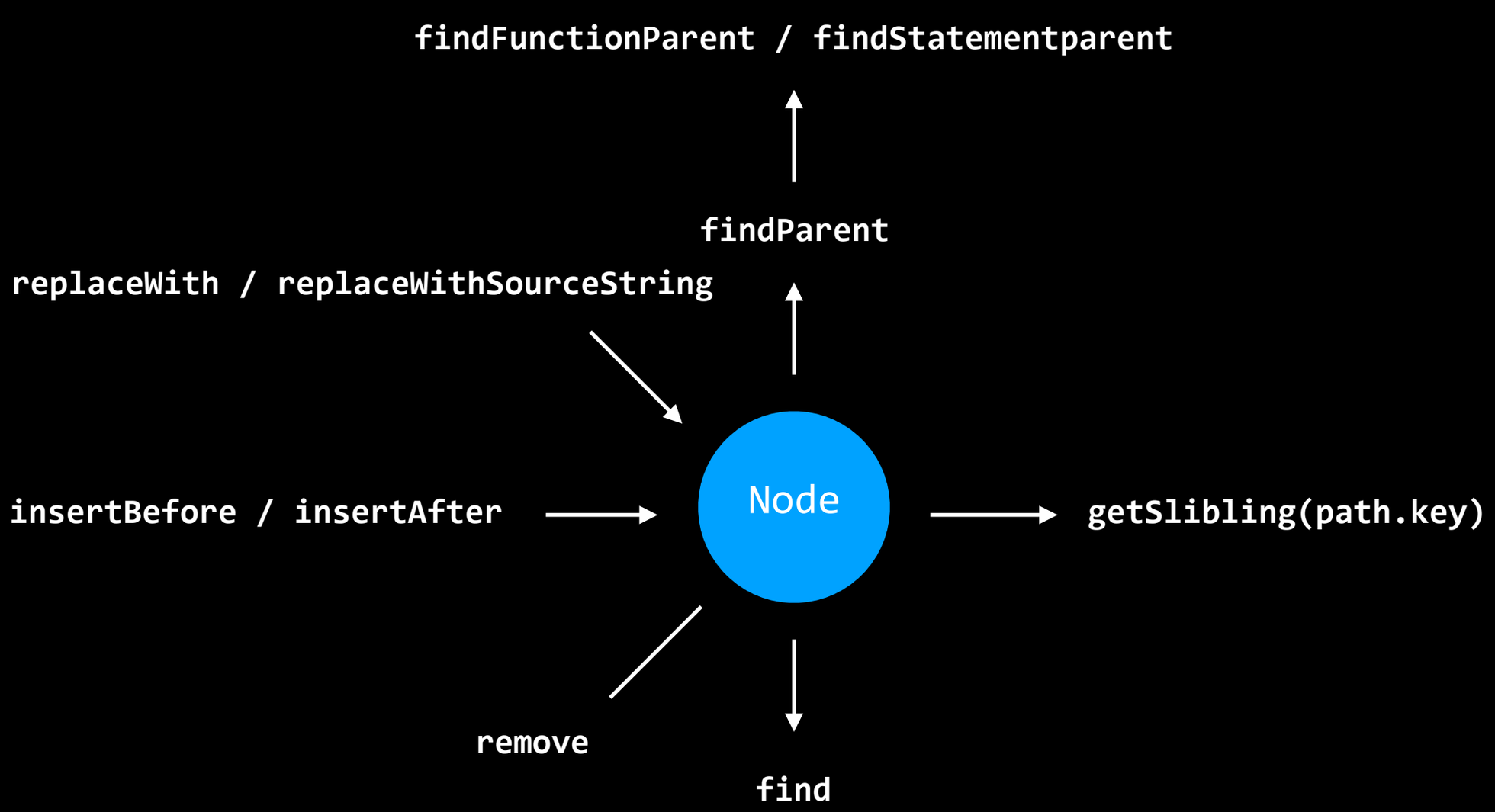
```
const types = require('babel-types');

function gAST_types() {
  const ast = types.returnStatement(
    types.binaryExpression(
      '+',
      types.identifier('x'),
      types.identifier('y')
    )
  );
  return ast;
}
```

```
const template = require('babel-template');

function gAST_template() {
  const tpl = template(`return VER + y;`);
  const ast = tpl({
    VER: types.identifier('x')
  });
  return ast;
}
```

Tools - path


findFunctionParent / findStatementparent
findParent
replaceWith / replaceWithSourceString
insertBefore / insertAfter
Node
getSlibling(path.key)
remove
find

webpack - 传参

```
{
  plugins: [
    [
      “path/to/plugin“,
      {
        "option1": true,
        "option2": false
      }
    ]
  ]
}
```

```
export default function({ types: t }) {
  return {
    visitor: {
      FunctionDeclaration(path, state) {
        console.log(state.opts);
        // { option1: true, option2: false }
      }
    }
  }
}
```

webpack - 报错

```
export default function({ types: t }) {
  return {
    visitor: {
      StringLiteral(path) {
        throw path.buildCodeFrameError("Error message here");
      }
    }
  };
}
```

```
file.js: Error message here
   7 |
   8 | let tips = [
>  9 |   "Click on any AST node with a '+' to expand it",
     |   ^
  10 |
  11 |   "Hovering over a node highlights the \
  12 |    corresponding part in the source code",
```

练习：代码出现变量 barret 则引入 barret 这个包。

```
let inserted = false;
traversal(ast, {
  Identifier: function (path) {
    if (path.node.name === 'barret' && !inserted) {
      const programPath = path.findParent((path) => path.isProgram());
      const container = programPath.get('body.0');
      inserted = true;
      container.insertBefore(template(`import barret from 'barret';`, {
        sourceType: 'module'
      })(), container);
    }
  }
});
```