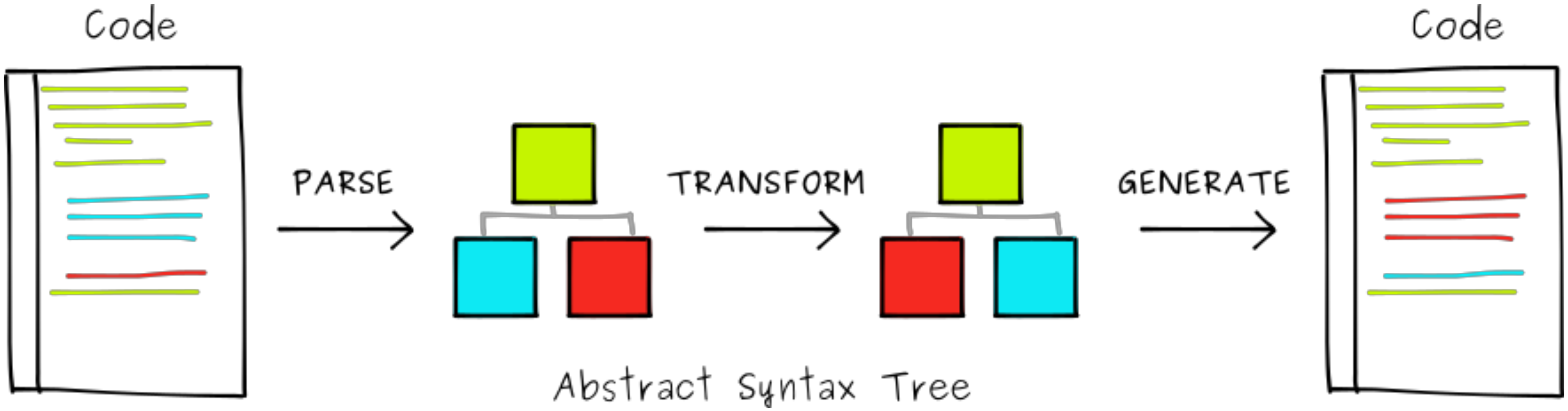
利用编译将Vue组件转成React组件

玉缜@闲鱼

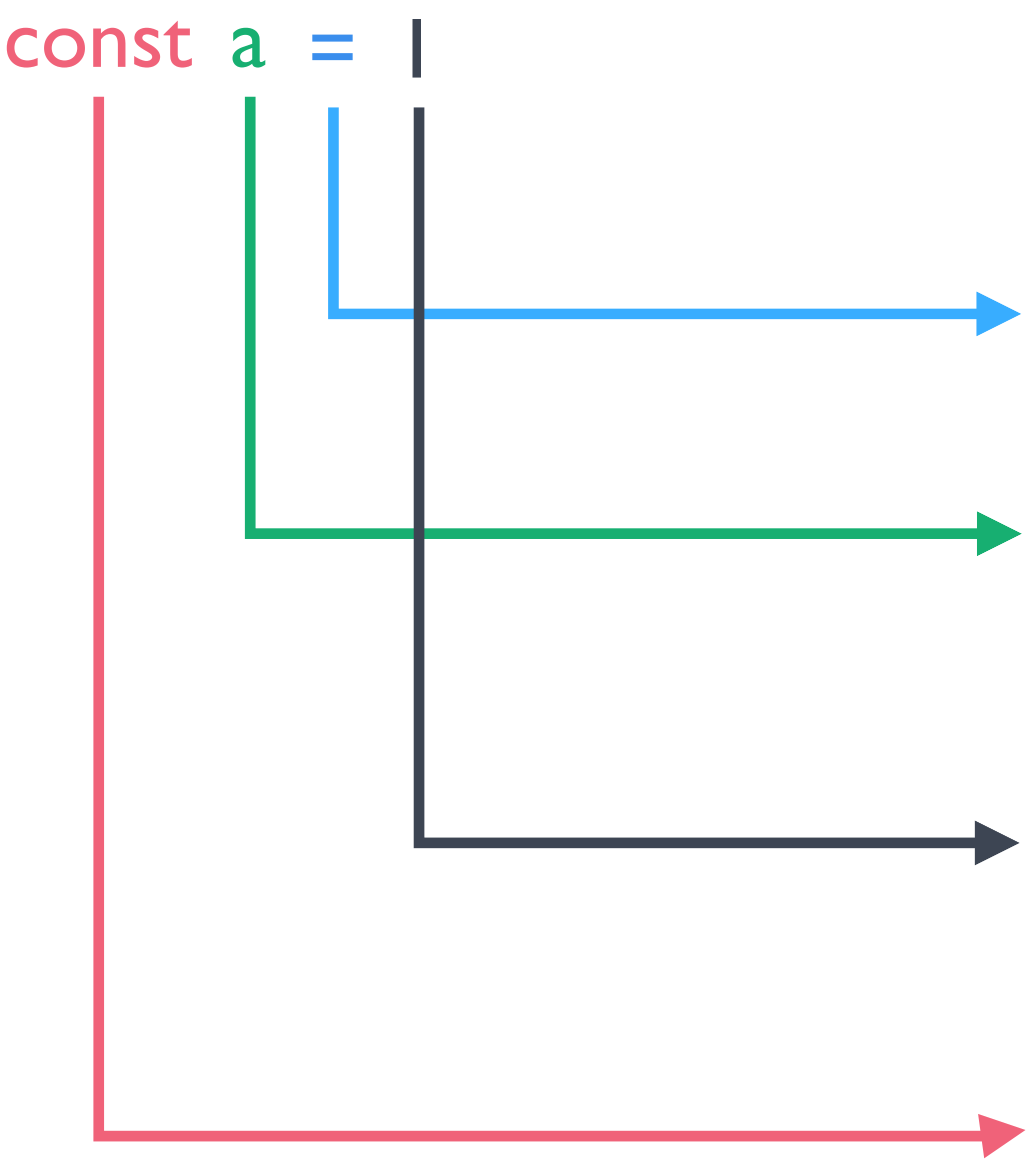

Code
PARSE
TRANSFORM
GENERATE
Code
Abstract Syntax Tree

const a = 1 → var a = 1

parse: 将const a = 1解析成AST

```
const parse = require('@babel/parser').parse;

const ast = parse('const a = 1');
```



```
{
  "type": "File",
  "program": {
    "type": "Program",
    "body": [{
      "type": "VariableDeclaration",
      "declarations": [{
        "type": "VariableDeclarator",
        "id": {
          "type": "Identifier",
          "name": "a"
        },
        "init": {
          "type": "NumericLiteral",
          "value": 1
        }
      }],
      "kind": "const"
    }]
  }
}
```

transform: 将const转换成var

```
const traverse = require('@babel/traverse').default;

traverse(ast, {
 VariableDeclaration: function(path) {
   if (path.node.kind === 'const') { //只有const的时候才处
理
     path.replaceWith(
       t.variableDeclaration('var', path.node.declarations)
     );
   }
   path.skip();
  }
});
```

generate: 转换后的AST生成代码

```
const generate = require('@babel/generator').default;

let code = generate(ast).code;
```

```
<template />

<style />

<script />
```

→

```
import 'index.css'

export default class Mod extends Component {
  ${transform(vue.script)}

  render() {
    ${transform(vue.template)}
  }
}
```

script

特殊属性处理

- data > state
- props > defaultProps & propTypes
- methods内函数属性提升
- components去除
- computed转成函数 show > this.show()

script

生命周期处理

- created > componentWillMount
- mounted > componentDidMount
- updated > componentDidUpdate
- beforeDestroy > componentWillUnmount

引用值处理

- 属性获取 this.show > this.state.show
- 属性更改 this.show=true > this.setState({show: true}))

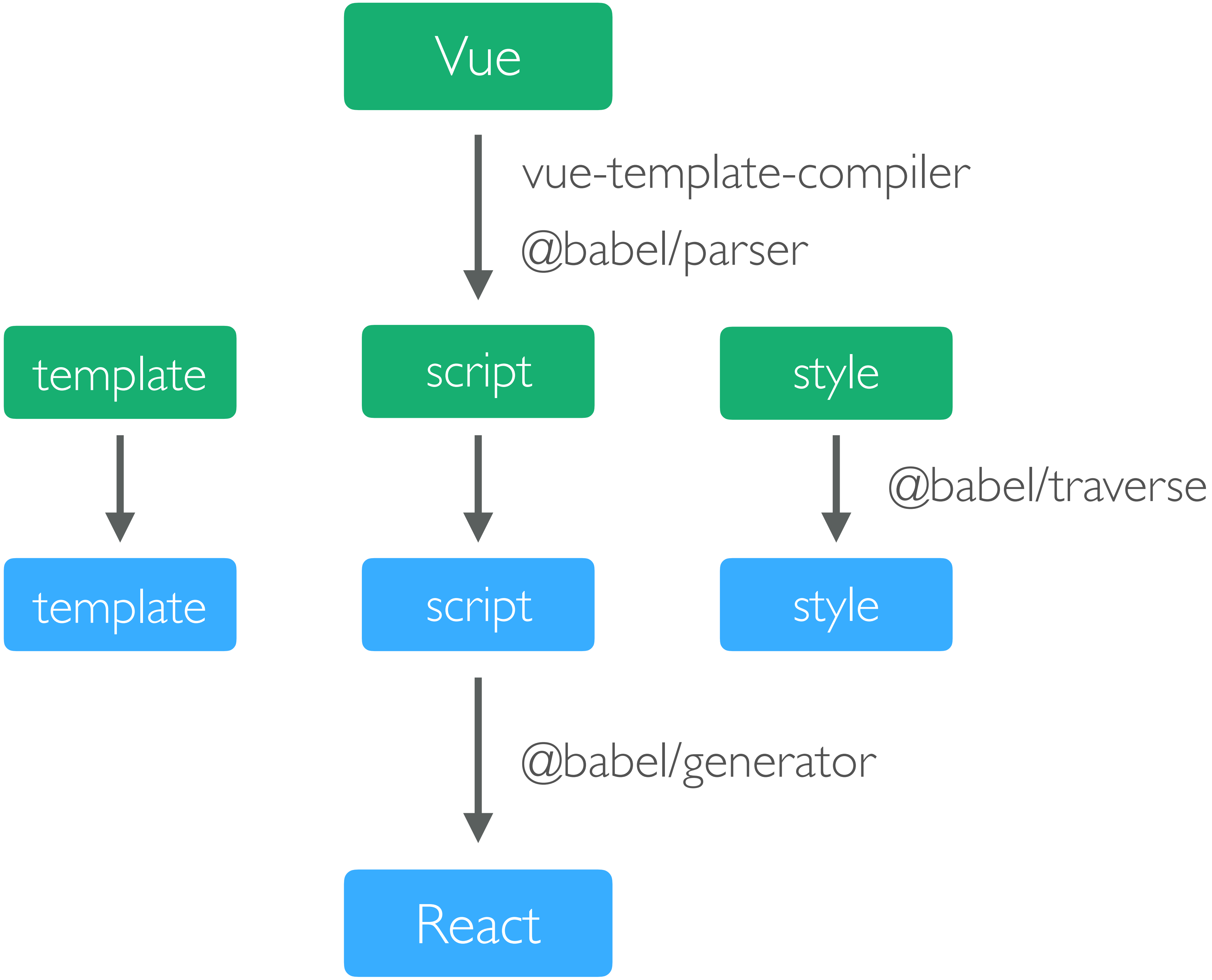
template

引用值处理

- {{title}} > {this.props.title}

增强指令处理

- 事件绑定 @click > onClick
- 逻辑判断 v-if="show" > {show && ……}
- 动态参数 :title="title" > title={title}
- 属性替换 class > className


Vue
vue-template-compiler
@babel/parser
template
script
style
@babel/traverse
template
script
style
@babel/generator
React

Parse Vue Data

```
data: {
  show: true,
  title: 'Vue2React'
}
```

→

```
data: {
  _statements, //data对象解析AST值
  show, //show属性解析AST值
  title //title属性解析AST值
}
```

Parse Vue

```
{
  app: {
    script: {
      ast, //script解析AST值
      components,
      computed,
      data: {
        _statements, //data对象解析AST值
        show, //show属性解析AST值
        title //title属性解析AST值
      },
      props,
      methods
    },
    style, // style字符串值
    template: {
      ast // template解析AST值
    }
  }
}
```

Transform React State

```
const t = require('@babel/types');

module.exports = (app) => {
  if (app.script.data && app.script.data._statements) {
    // classProperty 类属性
    // identifier 标识符
    // objectExpression 对象表达式
    return t.classProperty(t.identifier('state'), t.objectExpression(
      app.script.data._statements
    ));
  } else {
    return null;
  }
};
```

Transform v-if

```
if (ast.ifConditions && ast.ifConditions.length > 1) {
  // 包含v-else的情况
  let leftBlock = ast.ifConditions[0].block;
  let rightBlock = ast.ifConditions[1].block;

  let left = generatorJSXElement(leftBlock); //转化成JSX元素
  let right = generatorJSXElement(rightBlock); //转化成JSX元素

  child = t.jSXExpressionContainer( //JSX表达式容器
    // 转化成条件表达式
    t.conditionalExpression(
      parseExpression(value),
      left,
      right
    )
  );
}
```

Transform v-if

```
else {
  // 不包含v-else的情况
  child = t.jSXExpressionContainer( //JSX表达式容器
    // 转化成逻辑表达式
    t.logicalExpression('&&', parseExpression(value), t.jsxElement(
      t.jSXOpeningElement(t.jSXIdentifier(tag), attrs),
      t.jSXClosingElement(t.jSXIdentifier(tag)),
      children
    ))
  );
}
```

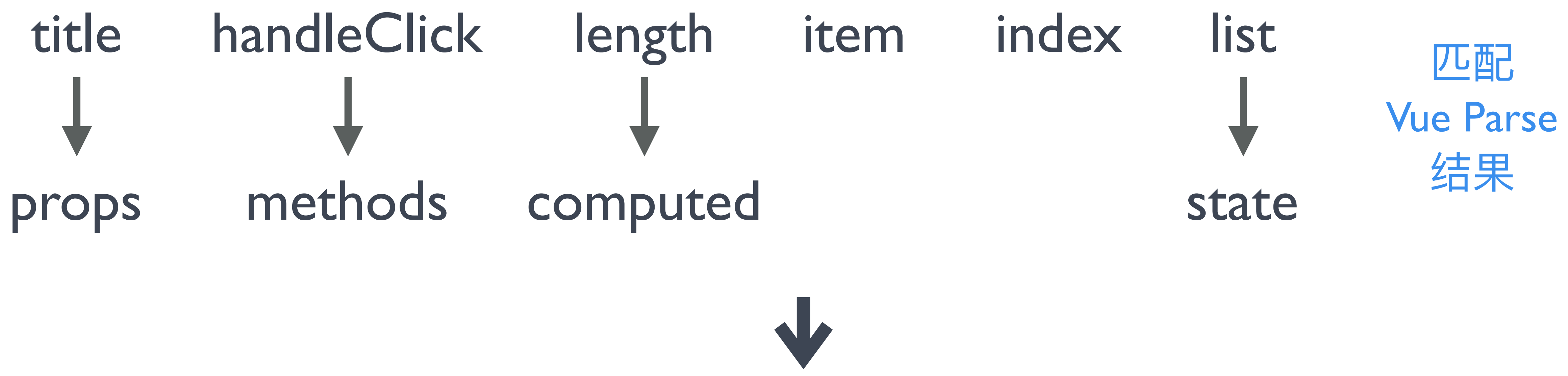
Transform Template Variable

```
<text class="title" @click="handleClick">{{title}}</text>
<text class="list-length">list length:{{length}}</text>
<div v-for="(item, index) in list" class="list-item" :key="`item-${index}`">
  <text class="item-text" @click="handleClick" v-if="item.show">{{item.text}}</text>
</div>
```

↓ 标识符提取

title   handleClick   length   item   index   list

Transform Template



title → props

handleClick → methods

length → computed

item

index

list → state

匹配
Vue Parse
结果

```
let {title} = this.props;
let {list} = this.state;
let {handleClick} = this;
let length = this.length();
```

Vue	Parse	Transform	React
style			**style**
script	提取AST 提取属性 (data/props/…)	转换属性 转换生命周期 处理引用属性	**script**
template	提取AST	处理增强指令 处理引用属性	**template**

What’s More

- 小程序转换
- 组件升级
- 打包插件
- ……

让你的闲置游起来