当前端遇上编译

目录

目录

- 编译原理在前端的应用
  - Babel
  - 工程化
  - 预处理
  - 模版引擎
  - 跨平台

编译原理与前端

解释型语言

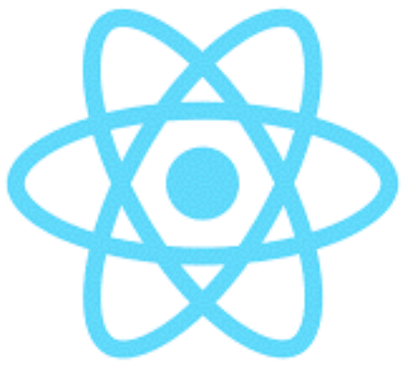
解释型语言（英语：Interpreted language），是一种编程语言。这种类型的编程语言，会将代码一句一句直接运行，不需要像编译语言（Compiled language）一样，经过编译器先行编译为机器代码，之后再运行。这种编程语言需要利用解释器，在运行期，动态将代码逐句解释（interpret）为机器代码，或是已经预先编译为机器代码的的子程序，之后再运行。

HTML

CSS

JavaScript

编译原理与前端

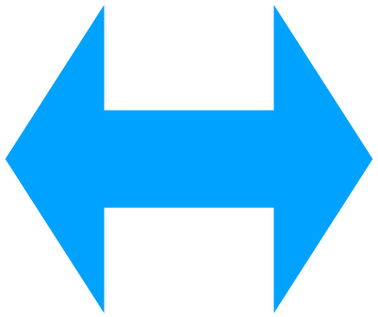


编译原理与前端

编译原理与前端

```
const code = "printf('hello world')";
const new_code = code.replace('printf', 'console.log');

eval(new_code);
```

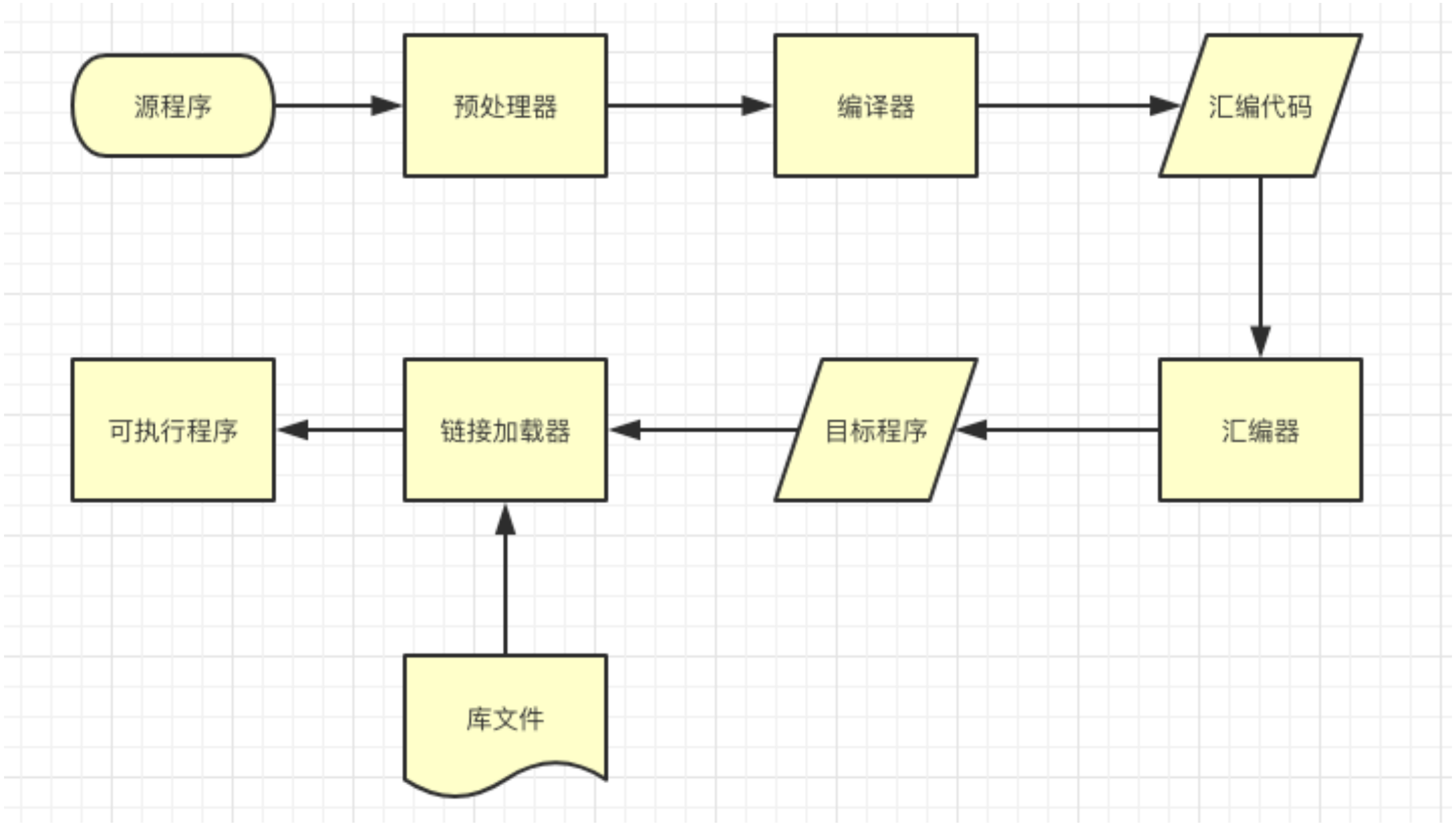
代码



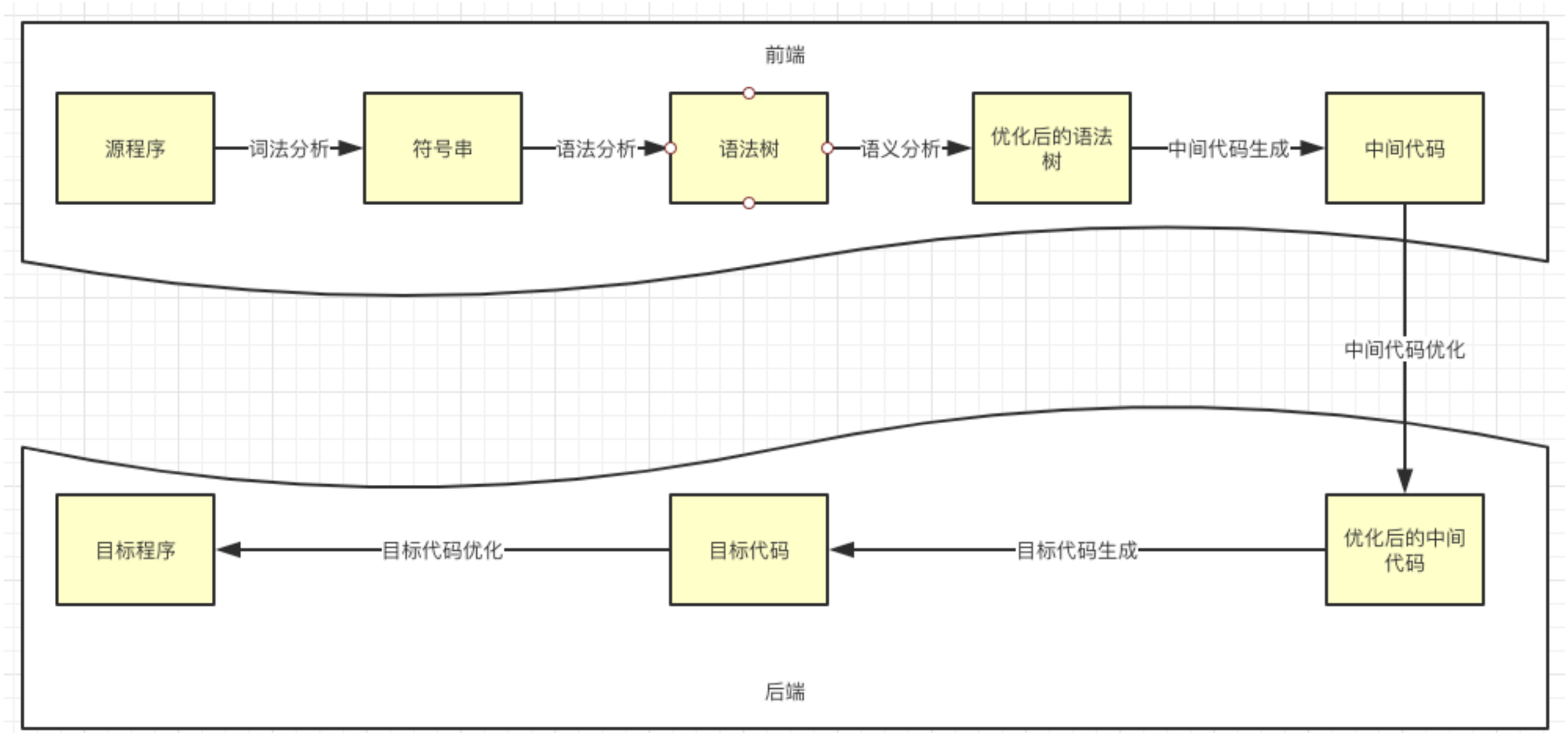
结构化的
文本数据格式

编译器

编译器



编译器


前端
源程序
词法分析
符号串
语法分析
语法树
语义分析
优化后的语法树
中间代码生成
中间代码
中间代码优化
目标程序
目标代码优化
目标代码
目标代码生成
优化后的中间代码
后端

编译器 - 词法分析

词法分析（英语：lexical analysis）是计算机科学中将字符序列转换为标记（token）序列的过程。 进行词法分析的程序或者函数叫作词法分析器（lexical analyzer，简称lexer），也叫扫描器（scanner）。

编译器 - 词法分析

(单词种类, 单词自身的值)

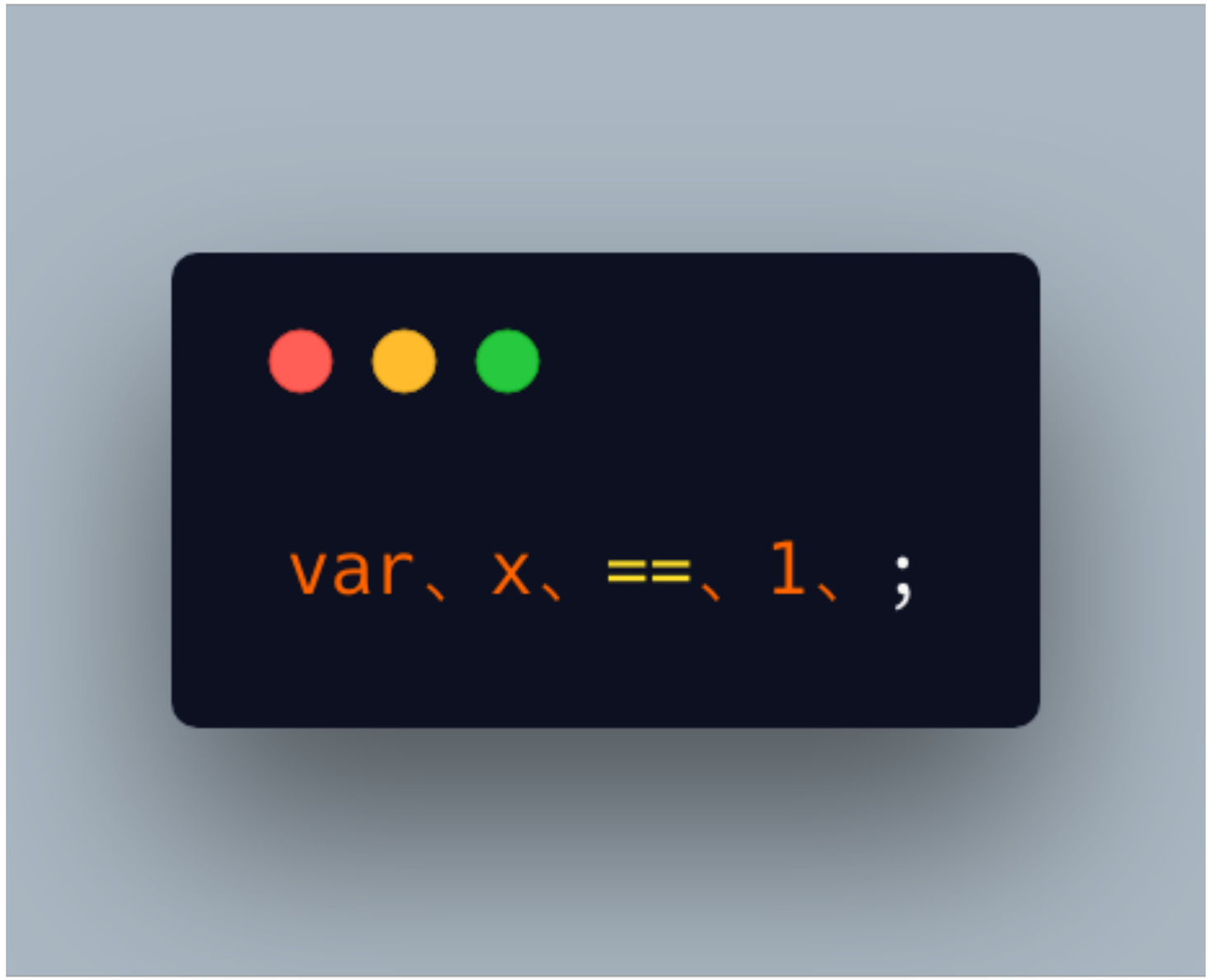
编译器 - 词法分析

```
const token = tokenizer('1 + 1');

//=> ['1', '+', '1']

// <=>

// [
//     {type: 'NUMBER', value:'1'},
//     {type: 'OPERATOR', value:'+'},
//     {type: 'NUMBER', value:'1'}
// ]
```

编译器 - 语法分析

语法分析（英语：syntactic analysis，也叫parsing）是根据某种给定的形式文法对由单词序列（如英语单词序列）构成的输入文本进行分析并确定其语法结构的一种过程。

编译器 - 语法分析

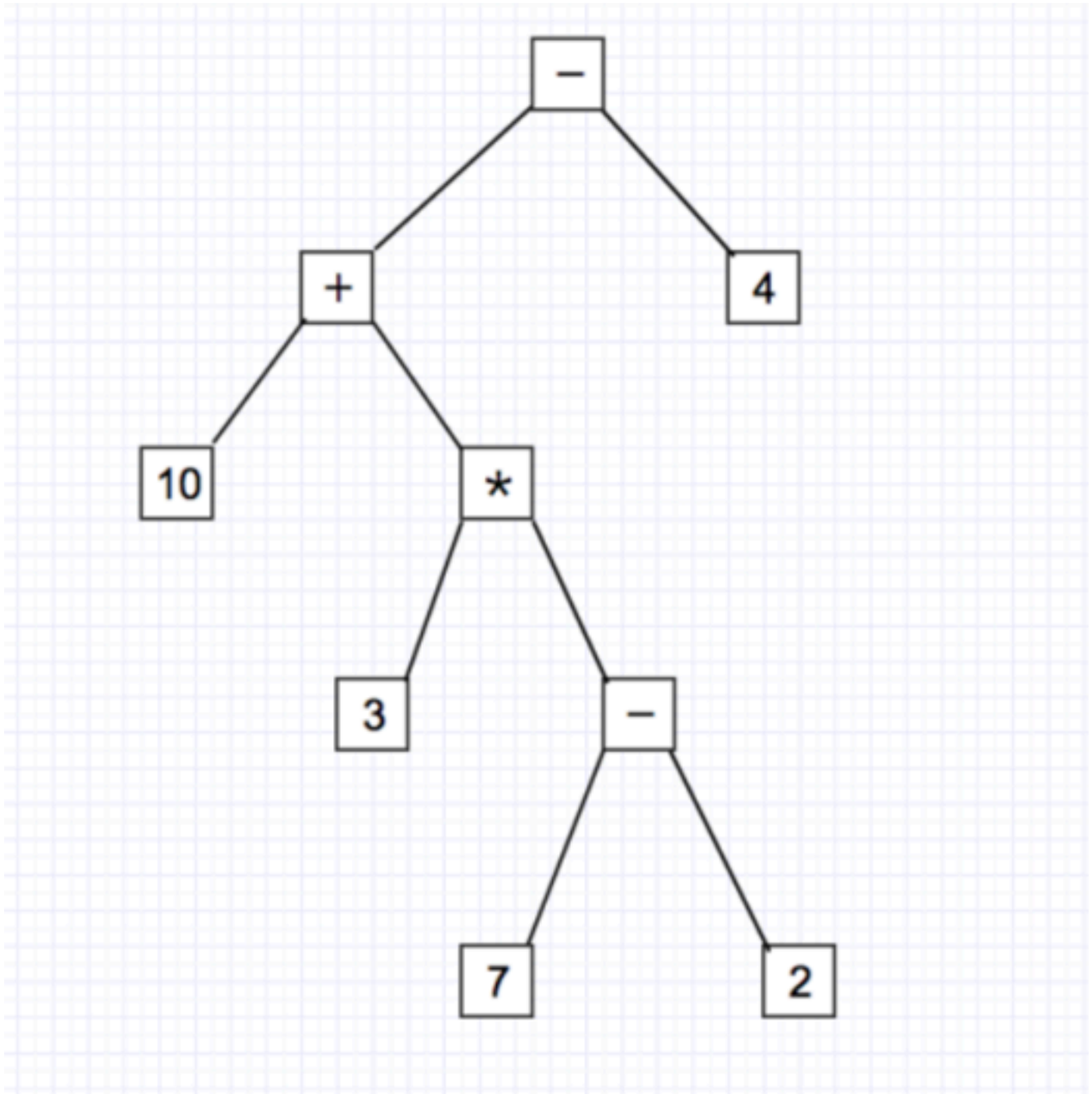


AST

抽象语法树，其实就是使用树状结构表示源代码的语法结构，树的每一个节点就代表源代码中的一个结构

编译器 - 语法分析

10+3*(7-2)-4



编译器 - 语法分析

AST是怎么生成的呢?

文法 + 结点类型

文法

程序设计语言的构造规则
用于指导整个语法分析的过程

一组终结符号
一组非终结符号
一组产生式
一个开始符号

编译器 - 语法分析

自顶向下

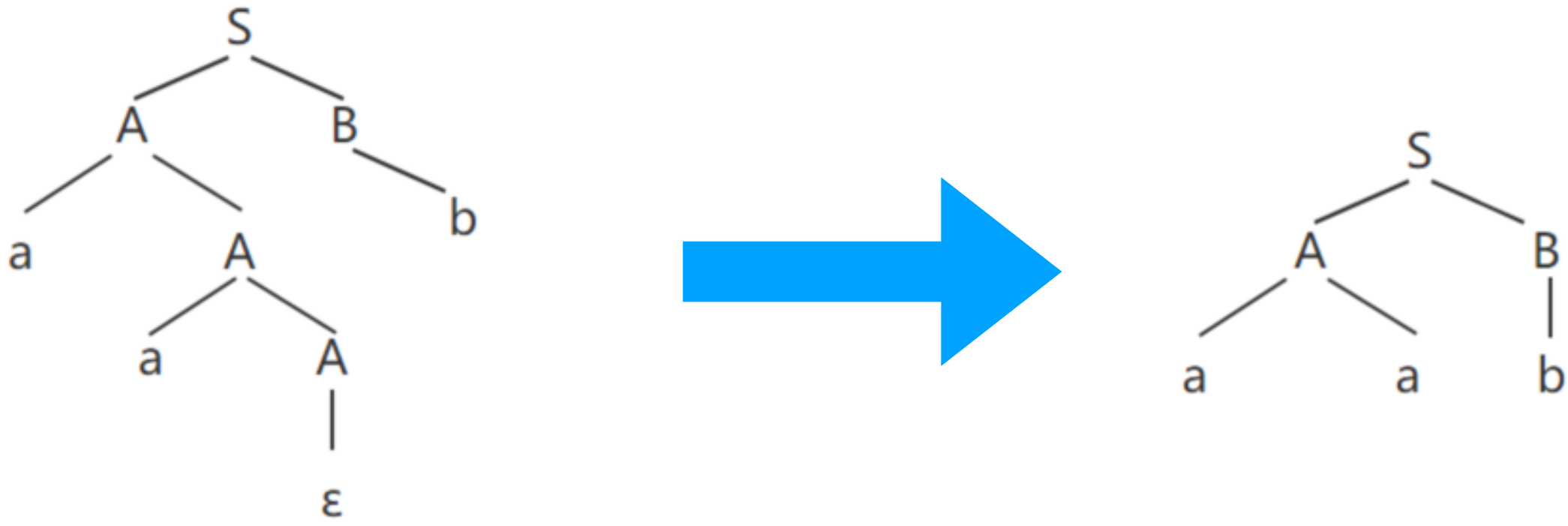
```
S -> AB
A -> aA | ε
B -> b | bB
```

aab

编译器 - 语法分析

```
S -> AB
A -> aA | ε
B -> b | bB
```

S => AB => aAB => aaAB => aaB => aab



编译器 - 语法分析

- 递归下降分析法
- LL(1)分析表
- 自底向上分析算法

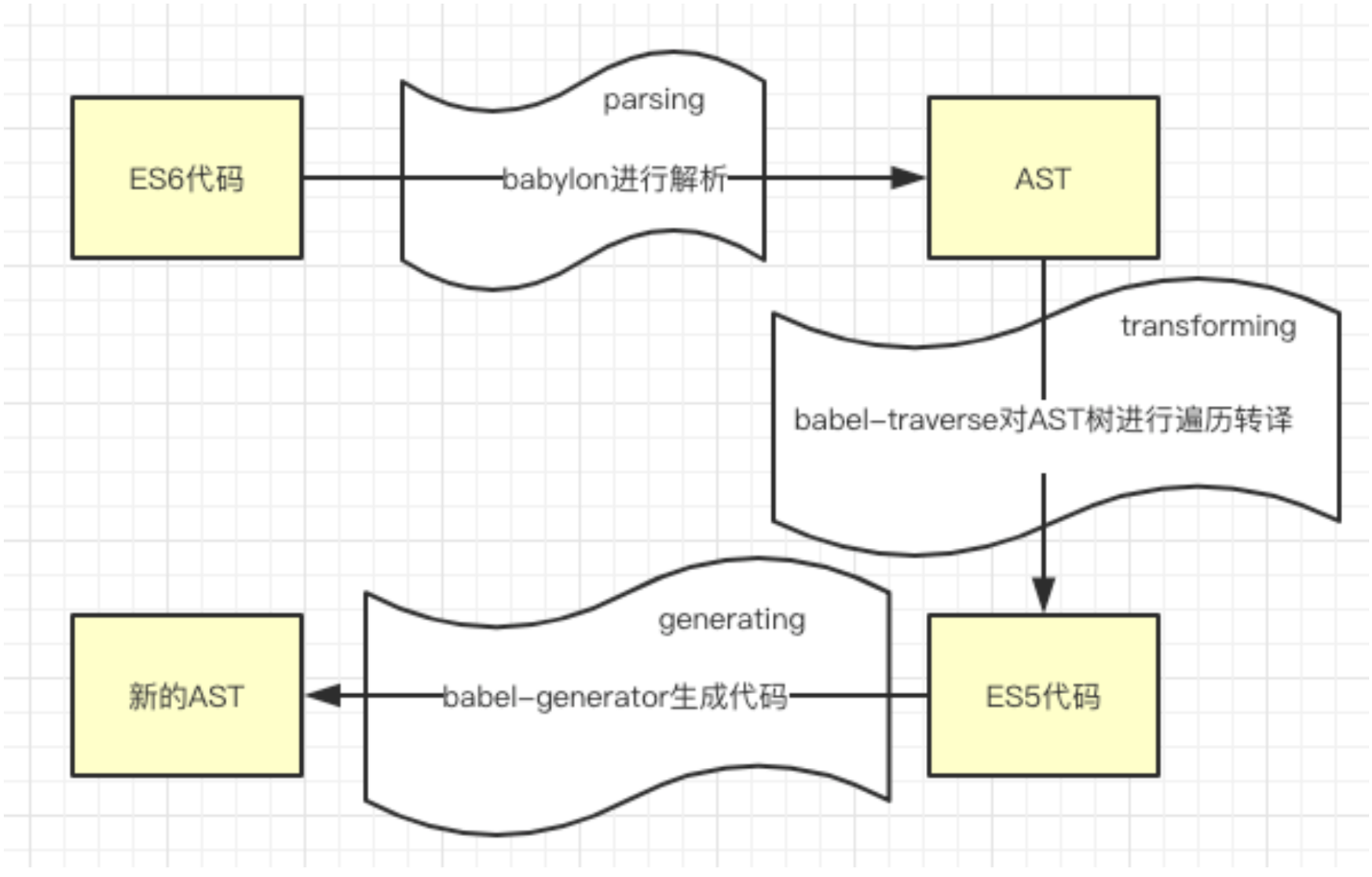
编译器 - 语法分析 - 作用

代码语法的检查
代码风格的检查
代码的格式化
代码的高亮
代码错误提示
代码自动补全等等

编译原理在前端的应用

编译原理在前端的应用

Babel



工程化

预处理

模版引擎

跨平台开发

QA

Thx