

极客大学前端进阶训练营

程劭非 (winter)

前手机淘宝前端负责人

重学JavaScript

编程语言基本知识

语言按语法分类

- 非形式语言
 - 中文，英文
- 形式语言（乔姆斯基谱系）
 - 0型 无限制文法
 - 1型 上下文相关文法
 - 2型 上下文无关文法
 - 3型 正则文法

产生式(BNF)

- 用尖括号括起来的名称来表示语法结构名
- 语法结构分成基础结构和需要用其他语法结构定义的复合结构
 - 基础结构称终结符
 - 复合结构称非终结符
- 引号和中间的字符表示终结符
- 可以有括号
- * 表示重复多次
- | 表示或
- + 表示至少一次

产生式(BNF)

四则运算:

- $1 + 2 * 3$

终结符:

- Number
- $+ - * /$

非终结符:

- MultiplicativeExpression
- AdditiveExpression

BNF

```
<MultiplicativeExpression> ::= <Number> |  
    <MultiplicativeExpression> "*" <Number> |  
    <MultiplicativeExpression> "/" <Number> |  
<AdditiveExpression> ::= < MultiplicativeExpression> |  
    <AdditiveExpression> "+" <MultiplicativeExpression> |  
    <AdditiveExpression> "-" <MultiplicativeExpression> |
```

练习：编写带括号的四则运算产生式

通过产生式理解乔姆斯基谱系

- 0型 无限制文法
 - $? ::= ?$
- 1型 上下文相关文法
 - $? <A> ? ::= ? ?$
- 2型 上下文无关文法
 - $<A> ::= ?$
- 3型 正则文法
 - $<A> ::= <A> ?$
 - $<A> ::= ? <A>$ ✖

```
{  
  get a { return 1 },  
  get : 1  
}
```

2 ** 1 ** 2

其它产生式

EBNF ABNF Customized

AdditiveExpression :
 MultiplicativeExpression
 AdditiveExpression +
 MultiplicativeExpression
 AdditiveExpression -
 MultiplicativeExpression

现代语言的特例

- C++中，* 可能表示乘号或者指针，具体是哪个，取决于星号前面的标识符是否被声明为类型
- VB中，< 可能是小于号，也可能是XML直接量的开始，取决于当前位置是否可以接受XML直接量
- Python中，行首的tab符和空格会根据上一行的行首空白以一定规则被处理成虚拟终结符indent或者dedent
- JavaScript中，/ 可能是除号，也可能是正则表达式开头，处理方式类似于VB，字符串模板中也需要特殊处理 }，还有自动插入分号规则

语言的分类

- 形式语言—用途

- 数据描述语言
- 编程语言

JSON, HTML, XAML, SQL, CSS

C, C++, Java, C#, Python, Ruby,
Perl,
Lisp, T-SQL, Clojure, Haskell,
JavaScript

- 形式语言—表达方式

- 声明式语言
- 命令型语言

JSON, HTML, XAML, SQL, CSS,
Lisp, Clojure, Haskell

C, C++, Java, C#, Python, Ruby,
Perl, JavaScript

练习：尽可能寻找你知道的计算机语言，
尝试把它们分类

图灵完备性

- 图灵完备性
 - 命令式——图灵机
 - goto
 - if和while
 - 声明式——lambda
 - 递归

动态与静态

- 动态：
 - 在用户的设备/在线服务器上
 - 产品实际运行时
 - Runtime
- 静态：
 - 在程序员的设备上
 - 产品开发时
 - Compiletime

类型系统

- 动态类型系统与静态类型系统
- 强类型与弱类型
 - String + Number
 - String == Boolean
- 复合类型
 - 结构体
 - 函数签名
- 子类型
- 泛型
 - 逆变/协变

```
{  
  a: T1  
  b: T2  
}
```

$(T1, T2) \Rightarrow T3$

凡是能用Array<Parent>的地方,
都能用Array<Child>

凡是能用Function<Child>的地方,
都能用Function<Parent>

一般命令式编程语言

Atom

- Identifier
- Literal

Expression

- Atom
- Operator
- Punctuator

Statement

- Expression
- Keyword
- Punctuator

Structure

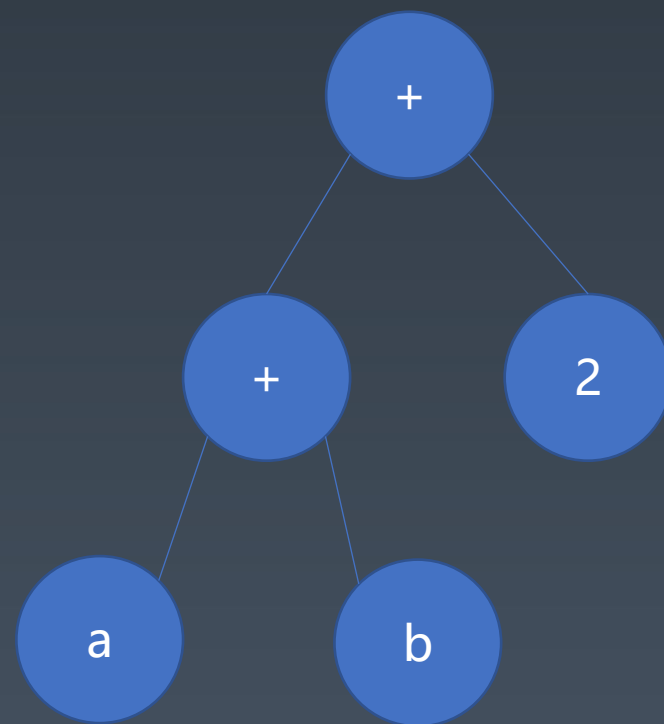
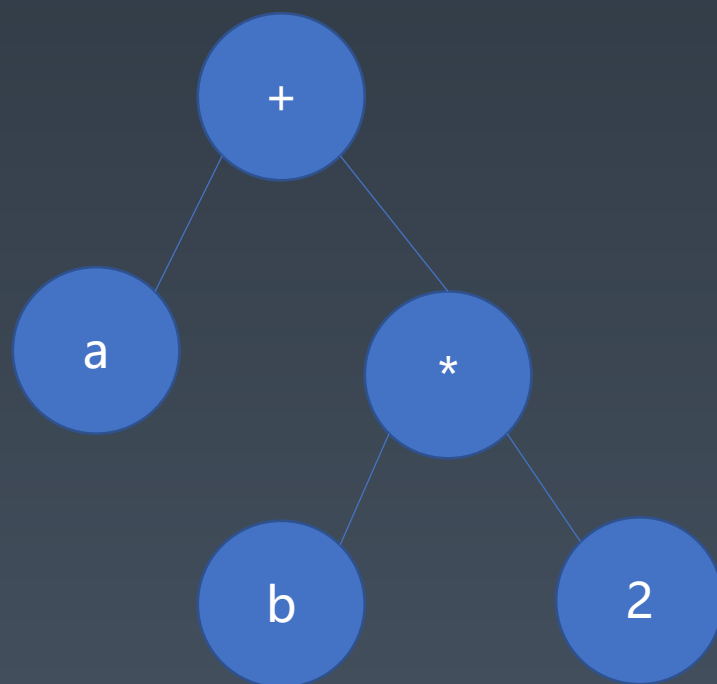
- Function
- Class
- Process
- Namespace
-

Program

- Program
- Module
- Package
- Library

重学JavaScript





THANKS! |  极客大学