Atitit 。AST的节点种类相对于解析树（parse tree）

AST的节点种类相对于解析树（parse tree）

通常要精简许多（括号、分号之类的token通常都扔掉了）

# 抽象语法树

[编辑](https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/javascript:;)

抽象语法树（abstract syn*tax tree*或者缩写为*AST*），或者**[语法树](https://baike.baidu.com/item/%E8%AF%AD%E6%B3%95%E6%A0%91/7031301" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)**（*syntax tree*），是[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)的抽象语法结构的树状表现形式，这里特指编程语言的源代码。和抽象语法树相对的是具体语法树（*concr*ete syntax*tree*），通常称作[分析树](https://baike.baidu.com/item/%E5%88%86%E6%9E%90%E6%A0%91" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)（*parse tree*）。一般的，在[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81/3969" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)的翻译和[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)过程中，[语法分析器](https://baike.baidu.com/item/%E8%AF%AD%E6%B3%95%E5%88%86%E6%9E%90%E5%99%A8" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)创建出分析树。一旦AST被创建出来，在后续的处理过程中，比如[语义分析](https://baike.baidu.com/item/%E8%AF%AD%E4%B9%89%E5%88%86%E6%9E%90" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)阶段，会添加一些信息。 [1]

在计算机科学中，**抽象语法树**（abstract syn*tax tree*或者缩写为*AST*），或者**[语法树](https://baike.baidu.com/item/%E8%AF%AD%E6%B3%95%E6%A0%91/7031301" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)**（*syntax tree*），是[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)的抽象语法结构的树状表现形式，这里特指编程语言的源代码。树上的每个节点都表示源代码中的一种结构。之所以说语法是“抽象”的，是因为这里的语法并不会表示出真实语法中出现的每个细节。比如，[嵌套](https://baike.baidu.com/item/%E5%B5%8C%E5%A5%97/3863565" \t "https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E8%AF%AD%E6%B3%95%E6%A0%91/_blank)括号被隐含在树的结构中，并没有以节点的形式呈现；而类似于if-condition-then这样的条件跳转语句，可以使用带有两个分支的节点来表示。

下图是hello.java中的“hello world“程序的AST示例:

