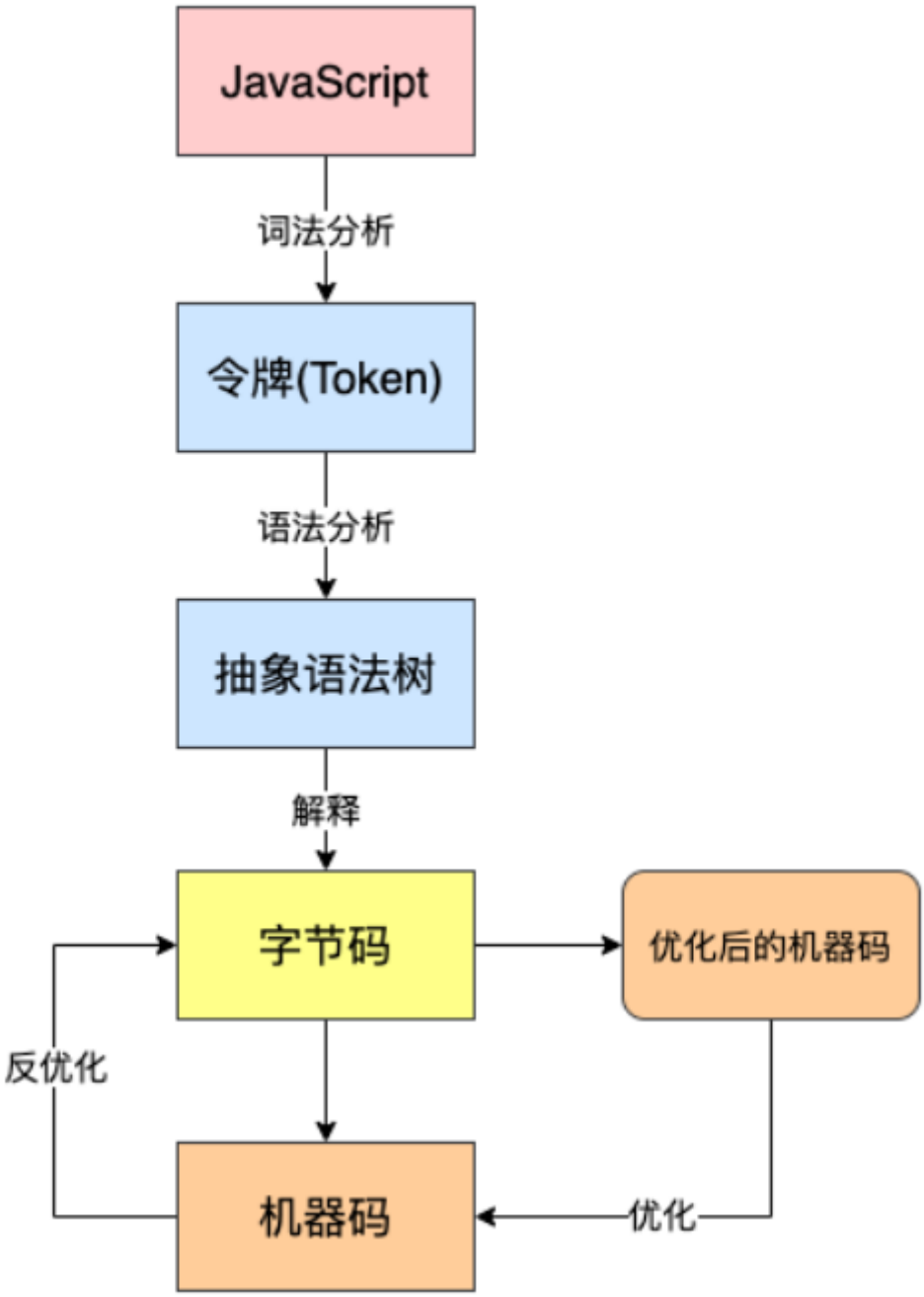
V8引擎如何执行Javascript代码

什么是字节码：

由于计算机不能直接运行我们写的代码，需要通过编译器和解释器将代码转换成机器码进行运行

- 字节码是介于AST和机器码之间的一种代码，字节码需要经过解释器将其转换成机器码之后才能执行
- 机器码所占用的空间远远超过了字节码，所以使用字节码可以减少系统的内存使用。

执行过程



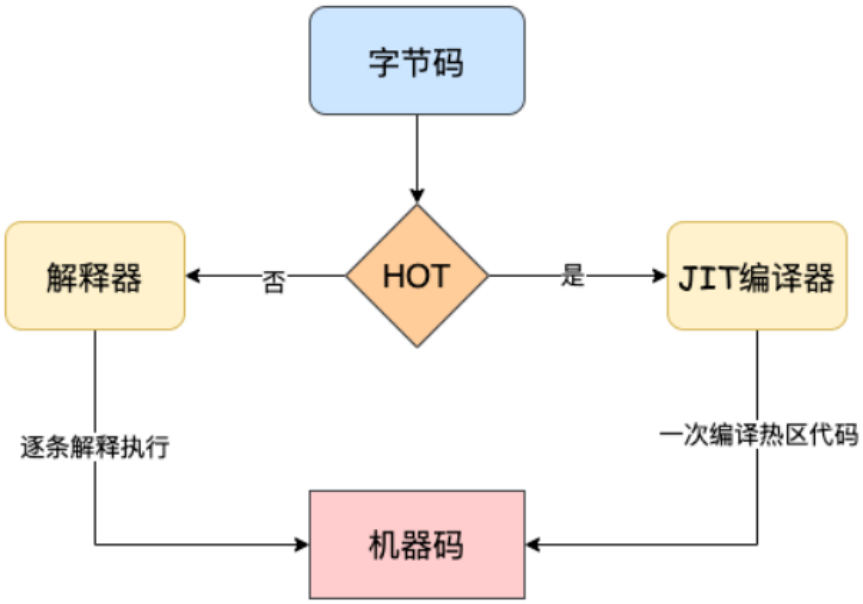
- **Parser解析器:** 负责将源代码转换成抽象语法树AST
  - **词法分析：**将源代码拆成最小的、不可再分的词法单元，称为 token。比如代码 var a = 1；通常会被分解成 var 、a、=1、;这五个词法单元。
  - **语法分析：**会用token生成一棵抽象语法树，生成树的过程中会去除不必要的符号令牌，然后按照语法规则来生成。
- **Ignotion解释器:** 根据抽象语法树生成字节码
- **Compiler编译器：** 将字节码转换成机器码

如果字节码是第一次执行，则解释器会逐条解析并运行

如果有**热代码**（重复执行的代码，运行次数超过某个阈值就被标记为热代码），后台的编译器就会把该段热点的字节码编译为高效的机器码，然后当再次执行这段被优化的代码时，只需要执行编译后的机器码

字节码配合解释器和编译器的技术就是 **即时编译（JIT）**。在 V8 中就是指解释器在解释执行字节码的同时，收集代码信息，当它发现某一部分代码变热了之后，编译器便闪亮登场，把热点的字节码转换为机器码，并把转换后的机器码保存起来，以备下次使用。

因为 V8 引擎是多线程的，编译器的编译线程和生成字节码不会在同一个线程上，这样可以和解释器相互配合着使用，不受另一方的影响。下面是JIT技术的工作机制：



V8引擎在处理JS过程中的一些优化策略：

- 函数只声明未被调用，不会被解析生成AST，也就不会生成字节码
- 函数只被调用一次，字节码直接被解释执行，编译器不会进行优化编译
- 函数被调用多次，可能会被标记为热点函数，可能会被编译成机器代码。当Ignition解释器收集的类型信息确定后，这是编译器则会将字节码编译为优化后的机器代码，以提高代码的执行性能，最后执行这个函数时，就直接运行优化后的机器代码。