1. V8环境安装
2. 首先安装gclient
3. 在gclient中安装 git
4. 使用 git 安装 dept\_tools

git clone <https://chromium.googlesource.com/chromium/tools/depot_tools.git>

4、然后执行gclient 执行gclient 用于 update gclient ， 执行gclinet sync ，并且安装gclient 的环境 如： python

5、执行 gclient config <https://chromium.googlesource.com/v8/v8.git> 安装V8环境

安装完V8环境后使用命令

d8 --print--byteCode simple.js

1. V8 是C++的编译库，可以单独用做C++库在服务器上运行，node的运行环境也是V8
2. V8的执行顺序，内部组成

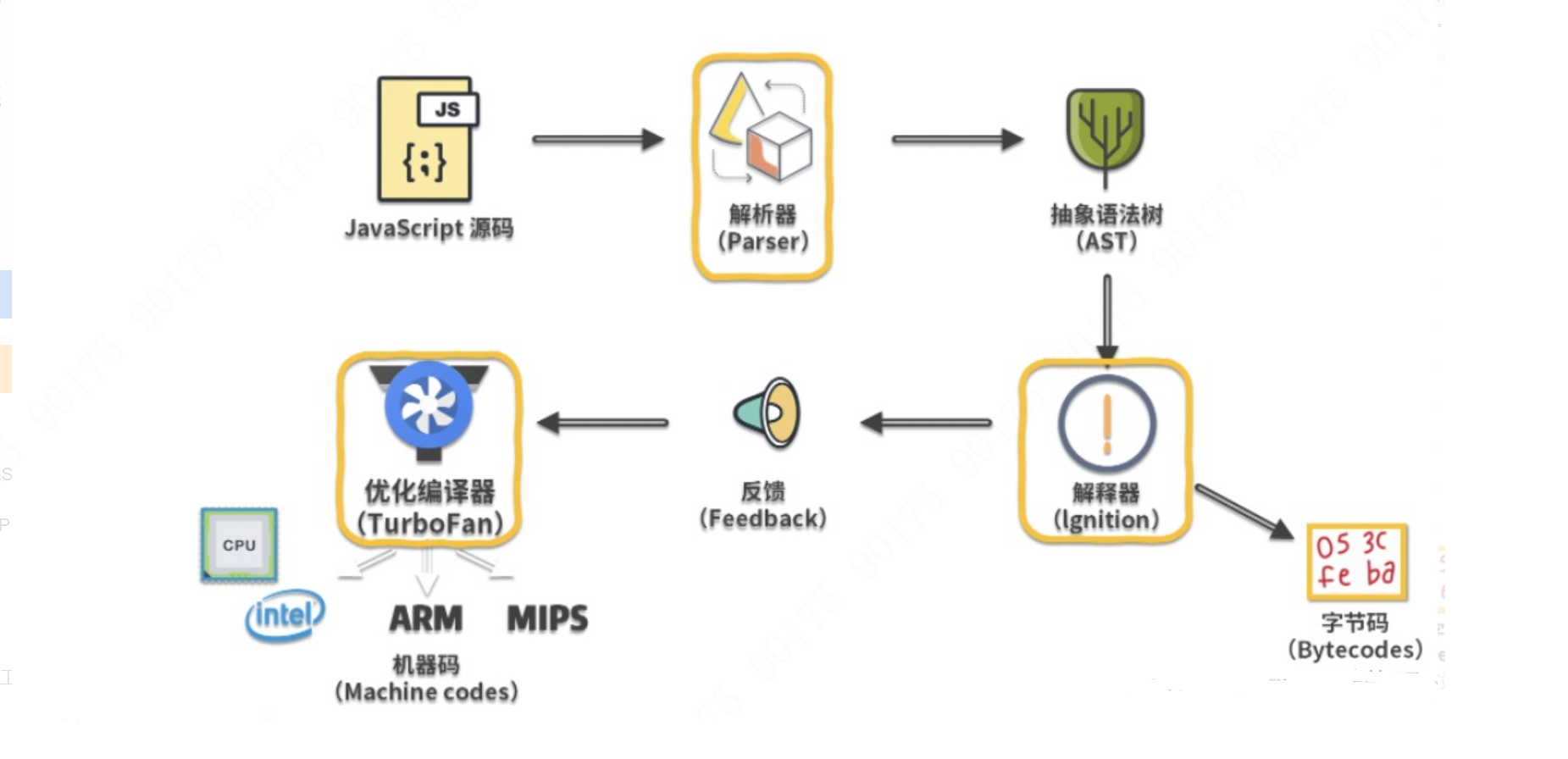
1、首先V8会将 源码转换成AST(抽象语法树)，

1、js源码 -> 【解析器】 -> AST(抽象语法树)

https://esprima.org/demo/parse.html

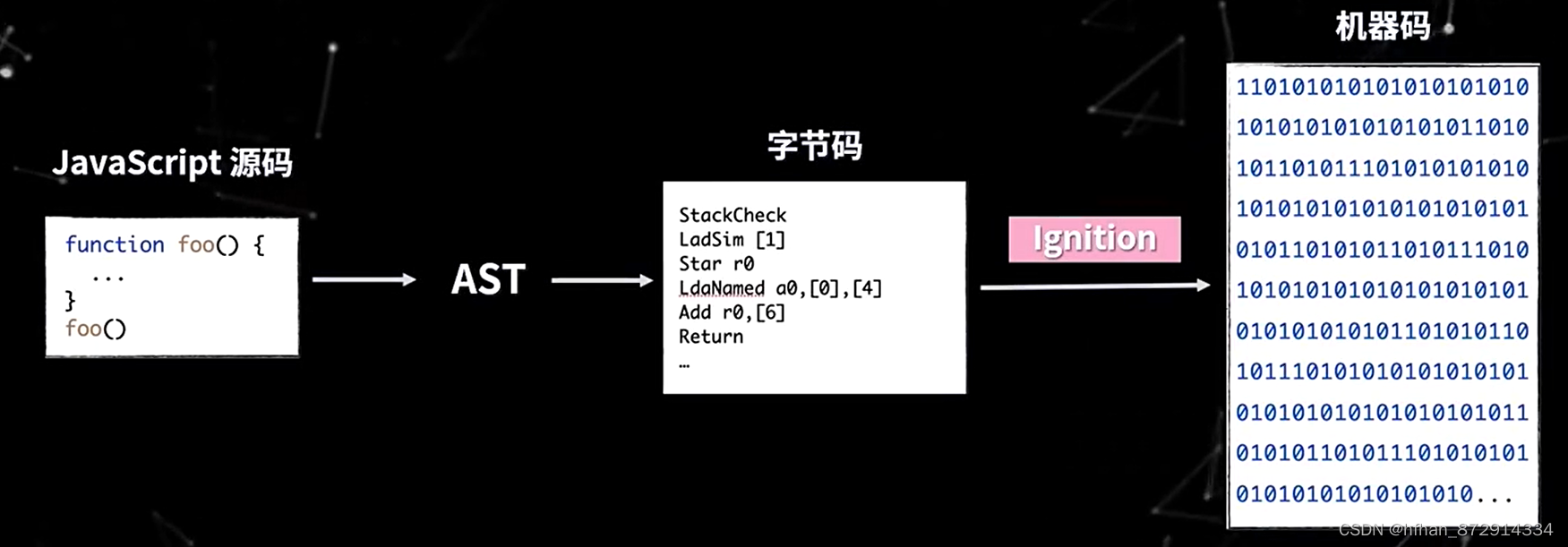
2、AST(抽象语法树) - 【解释器】-> 字节码 同时会编译成机器可以执行的机器码，这时抽象语法树就可以清除掉，释放缓存

3、由解释器逐行执行字节码，遇到热点代码启动编译器进行编译，生成对应的机器码, 以优化执行效率，同一个函数执行多次后，才会有优化编译，第一次执行都是根据当前函数之间编译

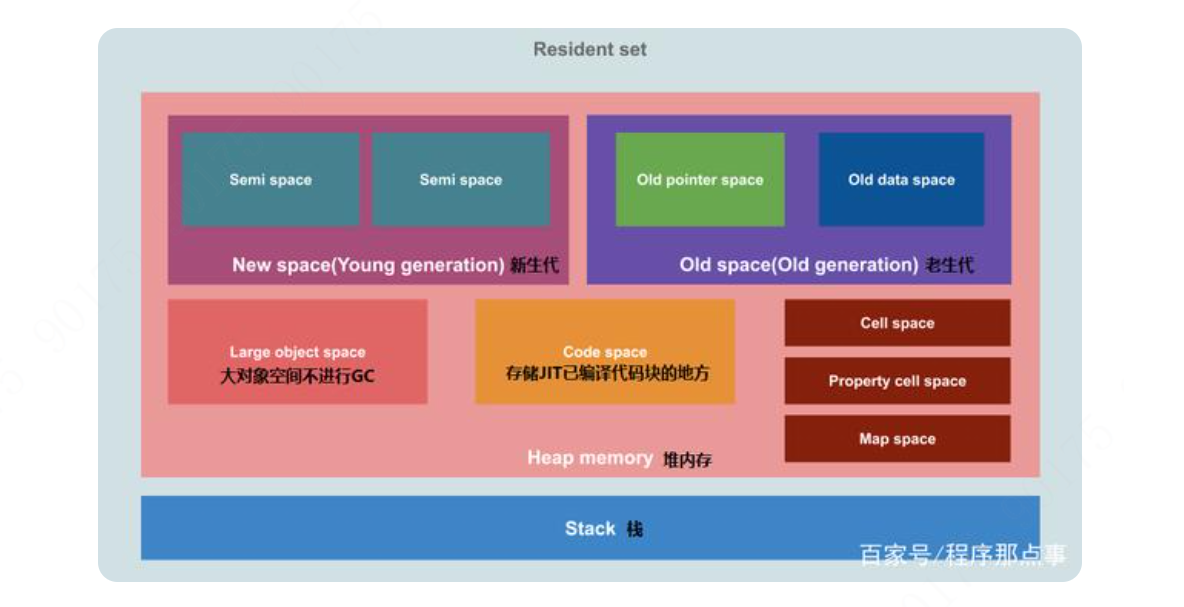


2、V8通过 解析器Ignition根据AST生成字节码并执行。

3、在这个过程中会收集反馈信息，交给TurboFan进行优化编译。TurboFan根据Ignition收集的反馈信息，将字节码编译为优化后的机器码，后续Ignition有优化后的机器码代替字节码执行。



1. 栈、堆 垃圾回收机制



1. 栈：一种受限制的线性表；(在操作过程中，并不是随意操作的，需要遵循特定的原则)
2. 线性表：线性表示最基础、简单的数据结构
3. 线性表的一对一逻辑关系
4. 先进先出 后进后出
5. 基本数据类型存储在栈内存中，引用数据类型存储在堆内存中，但是引用数据类型会在栈内存中存储一个实际对象的引用

全局执行上下文：默认的一个程序只有一个全局执行下文

每调用函数的时候，都会为这个函数创建一个新的执行上下文，每一个函数有一个执行上下文，一个程序中可以存在多个执行上下文，

函数的调用和执行就是压栈和出栈，递归除外

1、每次调用的时候会用 栈 内存去调用，然后去执行

2、定制的函数以字符串的形式存入到 堆 空间中

堆：

理论上，64位操作系统有 1.4G的内存（最新的nodeV14有2G）

64为新生代空间为 64MB，老生代空间为1400MB

1、新生代带的内存分为两个部分， from32MB to 32MB， 每次执行的时候总有一半的内存空置在新生代的内存中（空间换时间），当from中的内存已经满的时候，如果再存入新的内容到 新生代中，那么执行步骤是，先执行 复制当前还在使用的obj 到 to，将新的内存存入到 to中，然后再将from的内容全部清除，form变成to ，to变成from

2、当遇到特变大的obj 的时候不会存入到 新生代的内存中，也不会存入老生代的内存中，会单独生成一个大obj的内存，对象的数据类型是有限制的：必须是基础类型的数组，或者是string类对象。https://cloud.tencent.com/developer/article/1993463

1. 不同版本的编译器会有不同的规则，一般在执行 两次新生代交换后并且大于8Mb 后，会将新生代内存中的内容转到老生代，老生代的内容一般不会删除和改变，或者当对象超过新生代内存堆的时候会直接存入到堆中 ，原文件在 heap-inl.h中

https://gitee.com/zhufengpeixun/v8/blob/master/src/heap/heap-inl.h

1. 老生代 的内存清理方
2. 法 分 清除标记 标记整

理清除

为 标记清除 、标记整理，在执行垃圾回收的过程中，js脚本需要暂停，这种叫做stop the world

1. 由于老生代的内存会比较大所以一般会使用 requesrdleCallback ，切块执行，GC过程与主进程交替进行

增量标记法

开始标记---增量标记法----增量标记法-------清除---整理

惰性清理

并发、并行

堆中处理 500MB左右的数据垃圾大概时间为50ms

新生代 from to 老生代 1400M

扩展的内存是C++的内存

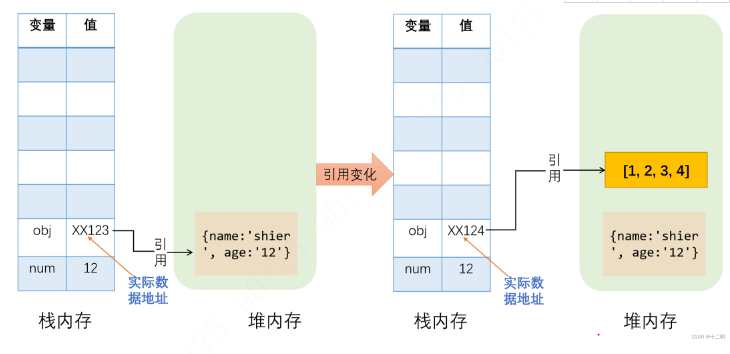
1、变量声明的本质是变量名与栈内存地址进行绑定，不直接与堆内存进行绑定。

2、声明的基本数据类型会将值存储在栈内存中，声明的复杂数据类型会将值存储在堆内存中并将其在堆中的内存地址作为值存到栈内存中。

3、const声明常量本质是指的是声明的变量名所指向的栈内存地址不可改变，但是栈中对应的值可以改变。

4、基本数据类型赋值是在栈内存中申请新的内存区域保存值并将其指向的内存地址绑定到原有变量上。

5、复杂数据类型赋值是在堆内存中申请新的内存区域保存值并将其指向的内存地址作为值在栈内存中申请新的内存区域保存将其在栈中的内存地址绑定到变量上。



1. 浏览器
   1. Js是单线程 ，浏览器是多线程
   2. 宏任务微任务，事件循环也是游离于js线程之外的
   3. 垃圾回收是游离于 js线程之外的
   4. window.performance 查看浏览器内存使用情况，
2. jsHeapSizeLimit: 4294705152 js堆内存大小限制
3. totalJSHeapSize: 116788133 表示当前js堆栈内存总大小
4. usedJSHeapSize: 103289773 表示所有被使用的js堆栈内存，usedJSHeapSize不能大于totalJSHeapSize，如果大于，有可能出现了内存泄漏。
   1. 想扩展内存，只能用更加底层的语言控制，C++
   2. 垃圾回收是浏览器的宏任务
5. 如避免缓存过大的问题
6. 少写console ，对于setTimeout等 要及时清除
7. 在new set new map中的 的变量 要逐级清除
8. 声明变量尽量一次性，因为V8会为每个对象分配一个隐藏类，如果对象结构发生变化，就会重建隐藏类，结构相同的对象会共享隐藏类
9. 尽量不要动态添加和删除属性

const int a = 1

int \*p = &a

\*p = 10

int \*p = (int \*) &a

相关资料

https://blog.csdn.net/qq\_37464878/article/details/121090738?spm=1001.2101.3001.6661.1&utm\_medium=distribute.pc\_relevant\_t0.none-task-blog-2~default~CTRLIST~Rate-1-121090738-blog-85501188.pc\_relevant\_multi\_platform\_whitelistv3&depth\_1-utm\_source=distribute.pc\_relevant\_t0.none-task-blog-2~default~CTRLIST~Rate-1-121090738-blog-85501188.pc\_relevant\_multi\_platform\_whitelistv3&utm\_relevant\_index=1

<https://blog.csdn.net/hf872914334/article/details/123891553>

<http://jsbench.me/>

<https://zhuanlan.zhihu.com/p/272519496>

<https://zhuanlan.zhihu.com/p/351623081>

<https://segmentfault.com/a/1190000038195578>

https://segmentfault.com/a/1190000021996331

https://blog.csdn.net/weixin\_34232744/article/details/88986890

https://blog.csdn.net/qiwoo\_weekly/article/details/118165529

https://blog.csdn.net/ed7zgee9x/article/details/85501188

https://blog.csdn.net/qiwoo\_weekly/article/details/118165529

https://blog.csdn.net/hf872914334/article/details/123891553

https://blog.csdn.net/kaimo313/article/details/118569519

https://blog.csdn.net/qq\_42543177/article/details/124644363

https://zhuanlan.zhihu.com/p/351623081

https://segmentfault.com/a/1190000021996331

https://zhuanlan.zhihu.com/p/25122691?utm\_medium=social&utm\_source=weibo

https://zhuanlan.zhihu.com/p/55903492

performance面板优化

# [Generator函数](https://www.cnblogs.com/songya/p/11694444.html)

<https://www.cnblogs.com/songya/p/11694444.html>

# **Web Worker**

https://zhuanlan.zhihu.com/p/396351215