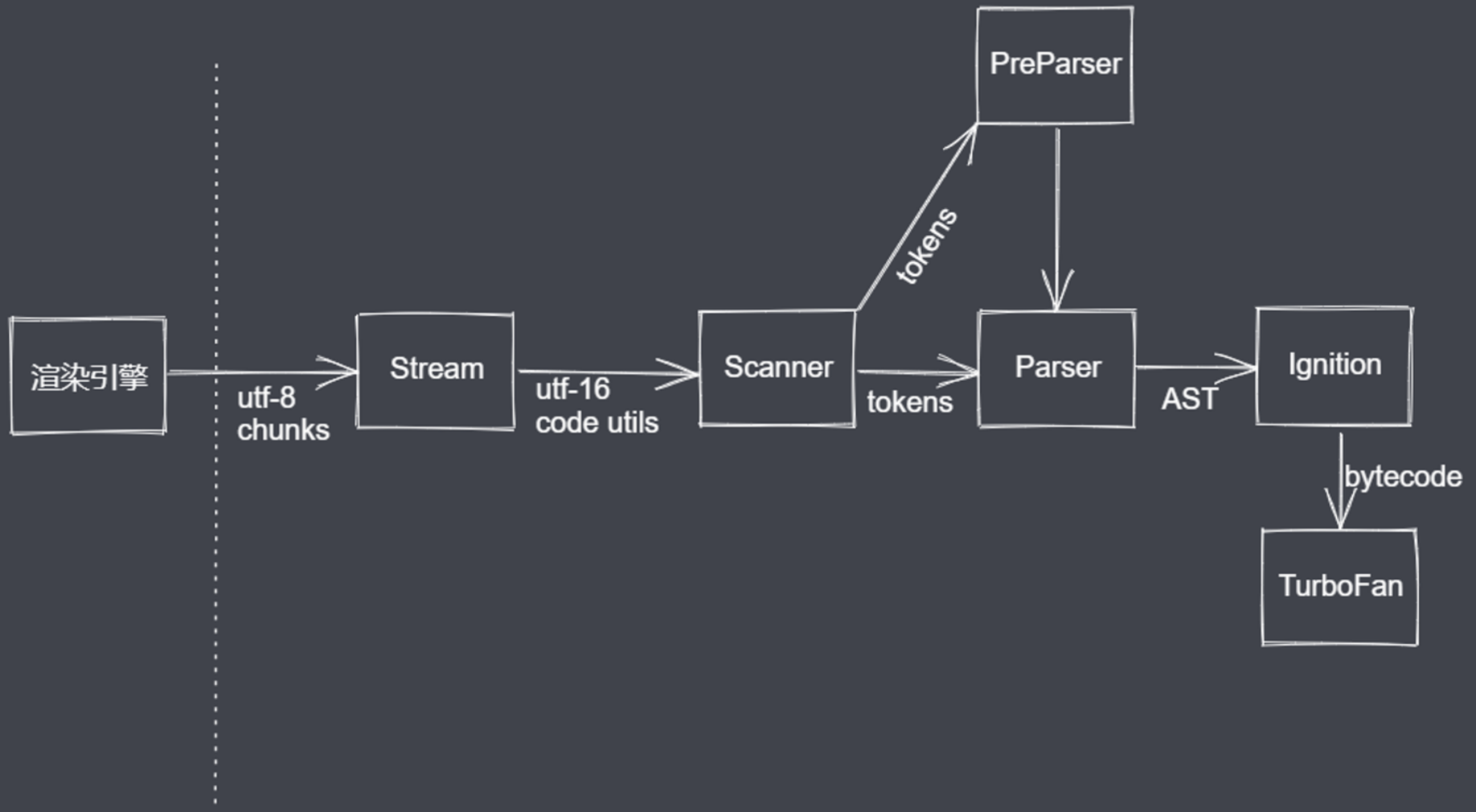
V8 引擎工作流程


渲染引擎
utf-8
chunks
Stream
utf-16
code utils
Scanner
PreParser
tokens
tokens
Parser
AST
Ignition
bytecode
TurboFan

Scanner 是一个扫描器

```
const username = 'alishi'
```

```
[
  {
    "type": "Keyword",
    "value": "const"
  },
  {
    "type": "Identifier",
    "value": "username"
  },
  {
    "type": "Punctuator",
    "value": "="
  },
  {
    "type": "String",
    "value": "alishi"
  }
]
```

Parser 是一个解析器

```
{
  "type": "Program",
  "body": [
    {
      "type": "VariableDeclaration",
      "declarations": [
        {
          "type": "VariableDeclarator",
          "id": {
            "type": "Identifier",
            "name": "username"
          },
          "init": {
            "type": "Literal",
            "value": 'alishi',
            "raw": "alishi"
          }
        }
      ],
      "kind": "const"
    }
  ],
  "sourceType": "script"
}
```

预解析优点

- 跳过未被使用的代码
- 不生成 AST， 创建无变量引用和声明的 scopes
- 依据规范抛出特定错误
- 解析速度更快

```
function func1() {
  console.log('func1')
}
function func2() {
  console.log('func2')
}

func2();
```

全量解析

- 解析被使用的代码
- 生成 AST
- 构建具体 scopes 信息，变量引用、声明等
- 抛出所有语法错误

```
// 声明时未调用，因此会被认为是不被执行的代码，进行预解析
function foo() {
  console.log('foo')
}

// 声明时未调用，因此会被认为是不被执行的代码，进行预解析
function fn() {}

// 函数立即执行，只进行一次全量解析
(function bar() {

})()

// 执行 foo，那么需要重新对 foo 函数进行全量解析，此时 foo 函数被解析了两次
foo()
```

Ignition 是V8 提供的一个解释器

TurboFan 是V8 提供的编译器模块

堆栈操作

堆栈准备

- JS 执行环境
- 执行环境栈（ECStack, execution context stack）
- 执行上下文
- VO(G)，全局变量对象

引用类型堆栈处理

函数堆栈处理

闭包堆栈处理

闭包与垃圾回收

循环添加事件

事件添加底层分析

事件委托

变量局部化

减少访问层级

缓存数据

防抖与节流

防抖函数实现

节流函数实现

拉勾教育

—互联网人实战大学—

扫码联系老师

技能评估、福利资料、课程优惠

Made with ❤ by LagouFed