variable✔

function declarations✔

blocks✔

conditionals statements and expressions✔

while loops->only for loop in Go✔

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Go | Source | JSON AST object in GP | JSOn AST object in HW |
| variable | var x = 3 | const x = 3; | 地图  描述已自动生成 | 文本  低可信度描述已自动生成 |
|  | var e,f int | N/A | 地图  描述已自动生成 |  |
|  | var a, b, c, d = "hi", 1, 1.12, true | N/A | 日程表  描述已自动生成 | N/A |
|  | f, g := "apple", "pine" | N/A | 图片包含 图表  描述已自动生成 | N/A |
| conditional | n := 0      if n > 0 {          n += 1      } else if n == 0 {          n += 1      } else {          n -= 1      } | let n = 0;  if (n > 0) {    n =n+ 1;  } else if (n === 0) {    n = 1;  } else {    n = -1;  } | 地图  描述已自动生成 | 日程表  中度可信度描述已自动生成 |
| for loop | package main  import "fmt"  func main() {  x := 0      i := 0      for i < 100 {          j := 0          for j < 100 {              x = x + i + j              j = j + 1          }          i = i + 1      }  } | N/A while loop | 图片包含 地图  描述已自动生成 |  |
| function declare& call | package main  import "fmt"  func fact(n int) int {      return factIter(n, 1, 1)  }  func factIter(n, i, acc int) int {      if i > n {          return acc      } else {          return factIter(n, i+1, acc\*i)      }  }  func main() {      result := fact(4)  } | let n = 0;  if (n > 0) {    n =n+ 1;  } else if (n === 0) {    n = 1;  } else {    n = -1;  } | 图片包含 日程表  描述已自动生成 |  |

Standard Project 2: Concurrent virtual machine for Go

The Concurrent virtual machine project applies the notion of a virtual machine (Module 4) to a concurrent programming language. Baseline expectations:

* Web-based implementation based on Source Academy frontend and js-slang (see [GitHubLinks to an external site.](https://github.com/source-academy))
* Implementation of a sublanguage of Go, consistent with a recent language specification
* Sequential language constructs: variable and function declarations, blocks, conditionals statements and expressions, while loops
* Concurrent constructs: Go routines, concurrency control (wait, wait group etc)
* Implementation should use a virtual machine
* Implementation must use a low-level memory model, and all runtime data structures need to be allocated from a single ArrayBuffer.
* Optional components:
  + visualization of heap and runtime stack
  + type checking
  + memory management