jeforthVM.js 相關資源 簡要說明

0. **exec(tib)** 外部源碼 解譯 **outer sourceCode interpreter**

```
var VM=new jeforthVM()
VM.type=function(msg){
  process.stdout.write(msg)
}
VM.exec('code . function(){print(" "+dataStack.pop())}end-code')
VM.exec('code cr function(){print("\n")}end-code')
VM.exec('5 . cr')
```

1. **tib** -- 外部源碼 內容
2. **iTib** -- 外部源碼 解譯位址
3. **waiting** -- 執意跳出迴圈 (單步 或 多功)
4. **error** -- 錯誤跳出迴圈
5. **resumeExec()** 外部源碼解譯 進入 迴圈

1. **call(ip)** 內部編碼 解譯 **inner compiledCode interpreter**

1. **compiledCode** -- 內部編碼 內容
2. **ip** -- 內部編碼 解譯位址
3. **returnStack** -- 回返堆疊, 主要 存放 回返上層呼叫指令 的 後續 ip
4. **waiting** -- 執意跳出迴圈 (單步 或 多功)
5. **error** -- 錯誤跳出迴圈
6. **resumeCall()** 內部編碼解譯 進入 迴圈

2. **dataStack** -- js 表列 (list) 當作 **資料堆疊** 在指令之間 傳遞資料

1. **dataStack=[]** -- 清空 資料堆疊 (預設)
2. **dataStack.push(t)** -- 將 t 放上 資料堆疊, t 可是 整數, 浮點數, 字串, 表列, 物件
   ...
3. **dataStack.length--** -- 若 資料堆疊 非空的, 丟掉最後放上的 資料
4. **t=dataStack[dataStack.length-1]** -- 讀取 最後放上 資料堆疊 的 t
5. **t=dataStack.pop()** -- 從 資料堆疊 取出 t

3. **returnStack** -- js 表列 (list) 當作 **回返堆疊** 存放 回返上層呼叫指令 的 後續 ip

1. **returnStack=[]** -- 清空 回返堆疊 (預設)
2. **returnStack.push(ip)** -- 將 ip 放上 回返堆疊
3. **t=returnStack[returnStack.length-1]** -- 讀取 最後放上 回返堆疊 的 t
4. **ip=returnStack.pop()** -- 從 回返堆疊 取出 ip

4. **base** -- js 變數 (variable) 解譯整數字串 或 印出整數字串 時的 **整數進制基數**

1. **base=10** -- 十進制 (預設)
2. **base=16** -- 十六進制
3. **base=2** -- 二進制

5. **nextToken** -- js 涵式 (function) 取 隨後 以空格區隔的 **源碼字串**

1. **t=nextToken()** -- 取 以空格區隔的 隨後源碼字串 t
2. **t=nextToken('"')** -- 取 空格之後 到 雙引號 " 之間的 隨後源碼字串 t
3. **t=nextToken('end-code')** -- 取 空格之後 到 end-code 之間的 隨後源碼字串 t

6. **findWord** -- js 涵式 (function) 取 名稱字串 所對應的 **指令位址**

1. **z=findWord(t)** -- 取 字串 t 對應 最後定義 指令位址 z (若未定義, 值 undefined)

7. **compiledCode** -- js 表列 (list) 存放 **編碼** (高階指令所呼叫 指令 及 資料)

1. **compiledCode.length** -- 下個 編碼 的 存放位址
2. **compiledCode.push(w)** -- 將 編碼 w 加入 compiledCode

8. **newWord** -- js 涵式 (function) 定義 新 **低階指令**

1. newWord('**doLit**',function(){dataStack.push(compiledCode[ip++])})
2. newWord('**exit**',function(){ip=returnStack.pop()})

9. **compileCode** -- js 涵式 (function) 將 所呼叫 指令/資料 **編碼加入** compiledCode

1. compileCode('**doLit**',5) -- 編碼 將 doLit 及 5 加入 compiledCode
2. compileCode('**exit**') -- 編碼 將 exit 加入 compiledCode

10. **newWord** -- js 涵式 (function) 定義 新 **高階指令**

1. **xt=compiledCode.length**
2. compileCode('doLit',5),compileCode('exit')
3. newWord('five',**xt**)

11. **lastWord** -- js 變數 (variable) 指向 **最後定義的指令**

1. lastWord.**compileOnly**=1 -- 最後定義的指令 只能在編譯狀態下執行
2. lastWord.**immediate**=1 -- 最後定義的指令 也能在編譯狀態下執行

12. **vocs** -- js 表列 (list) 蒐集 所有的 **詞彙**

1. vocs[iv].**words** -- js 表列 (list) 蒐集 詞彙 vocs[iv] 中 所有的 指令
2. vocs[iv].**index** -- js 物件 (object) 詞彙 詞彙 vocs[iv] 中 指令的 定義位址 表列
3. vocs[**current**] -- 蒐集 新 指令 的 當前詞彙
4. vocs[**context**[i]] -- 搜尋 指令 的 指定詞彙 i = 0, 1 2 , ...