Google Developer Groups

Beijing

Excelize: Go 语言 WebAssembly 移植实践

续日

Excelize Maintainer

GitHub: @xuri

19 November, 2023

内容提要

1. 跨语言移植的应用与技术背景
2. Go 语言 WebAssembly 移植要点
    2.1 基本数据类型处理
    2.2 复合数据类型处理
    2.3 函数的移植方法
    2.4 编译优化
    2.5 发布 wasm 文件
3. 浏览器/Node.js 运行时应用 WebAssembly 案例

1. 跨语言移植的应用与技术背景

应用场景

一套代码复用在多个平台：

一致性要求高、处理需求单一、减少维护成本

利用其它语言生态库的能力：

多媒体编解码、办公文档格式解析

技术背景

交叉编译：为不同平台分别构建

静态链接库、动态链接库

虚拟机：字节码和机器码运行时

WebAssembly、JVM、脚本语言虚拟机

2. Go 语言 WebAssembly 移植要点

2.1 基本数据类型处理

2.2 复合数据类型处理

2.3 函数的移植方法

2.4 编译优化

2.5 发布 WASM 文件

Go 构建 WebAssembly 操作 DOM

```go
package main

import (
    "syscall/js"
    "time"
)

func myGoFunc(this js.Value, args []js.Value) interface{} {
    js.Global().Get("result").Set("innerHTML", args[0].String())
    return nil
}

func main() {
    done := make(chan int, 0)
    js.Global().Set("myGoFunc", js.FuncOf(myGoFunc))
    <-done
}
```

```html
<html>
...
<body>
    <button id="btn" onclick="myGoFunc('Hello')">Click</button>
    <p id="result"></p>
</body>
</html>
```

2.1 基本数据类型处理

JavaScript Datatypes	Go Datatypes
function	js.Func
[its value]	js.Value
null	nil
boolean	bool
number	int, in32, int64, uint, uint32, uint64, float, float32, float64, …
string	string
new array	[]interface{}
new object	map[string]interface{}

2.2 复合数据类型处理

一个 JavaScript 对象转 Go 变量的例子：

JavaScript Object

```
let A = {
    B: {
        D: 'hello'
    }
}
```

接收以 js.Value 数据类型表示的 JavaScript 对象

```
func jsToGo(jsVal js.Value) (A, error) {
    var a A
    jsA := jsVal.Get("A")
    if jsA.Type() != js.TypeUndefined {
        if jsA.Type() != js.TypeObject {
            return a, errors.New("type error")
        }
        jsB := jsA.Get("B")
        if jsB.Type() != js.TypeUndefined {
            if jsB.Type() != js.TypeObject {
                return a, errors.New("type error")
            }
            jsD := jsB.Get("D")
            if jsD.Type() != js.TypeUndefined {
                if jsD.Type() != js.TypeString {
                    return a, errors.New("type error")
                }
                a.B.D = jsD.String()
            }
        }
    }
    return a, nil
}
```

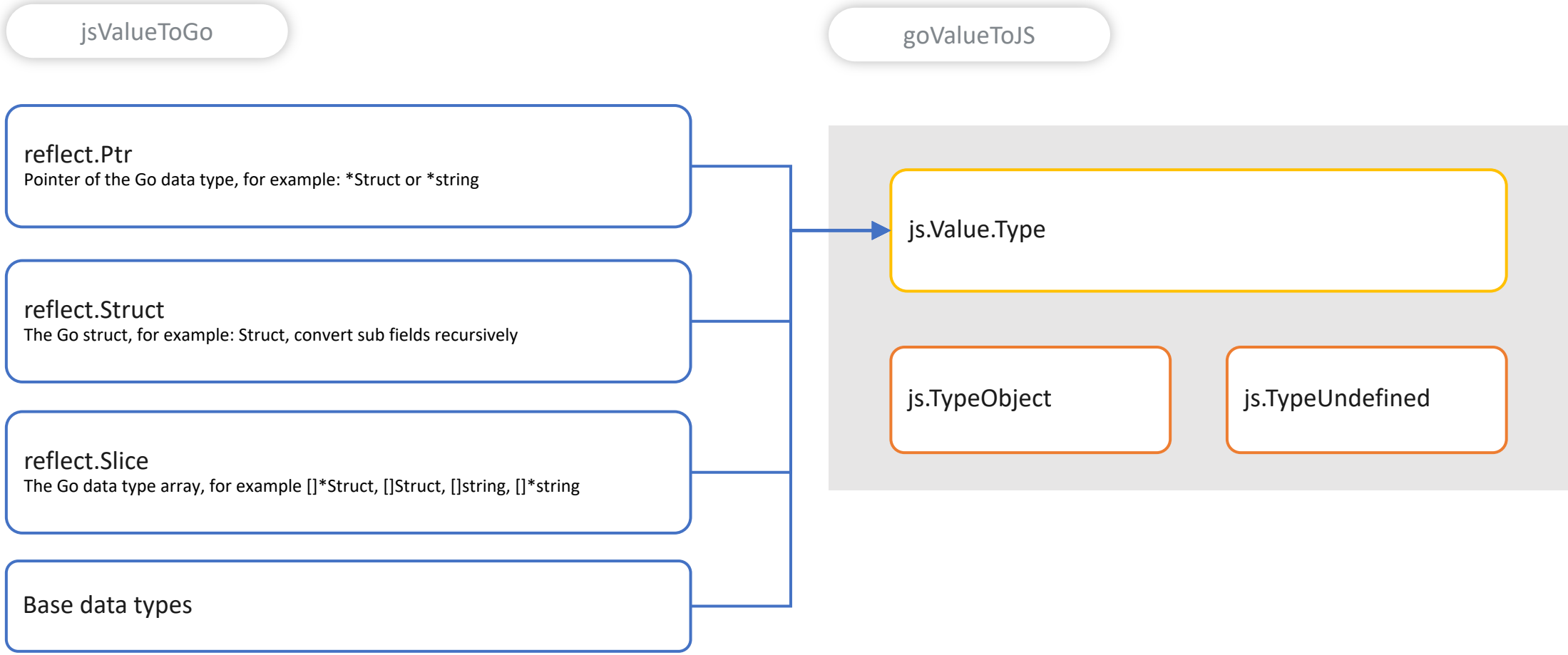
更简单的转换方法?

Go Data Structure

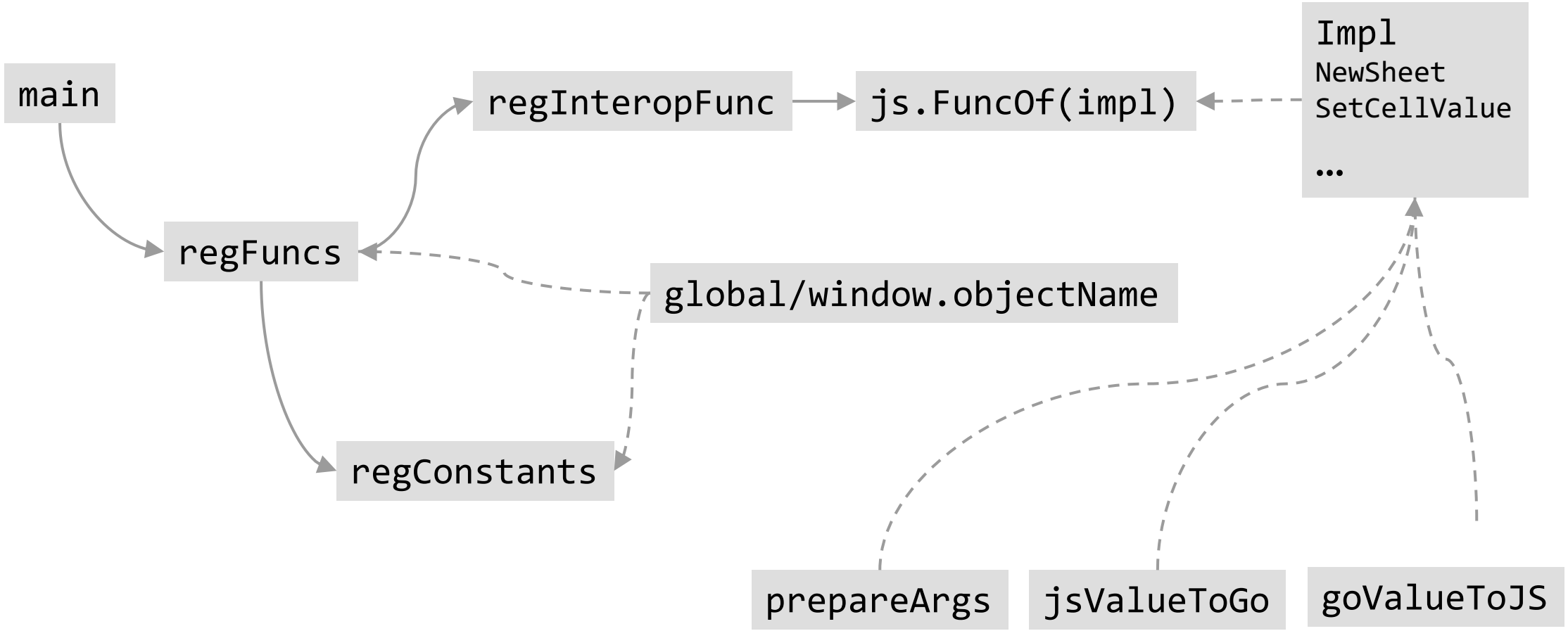
```
type A struct {
    B *C
}

type C struct {
    D string
}

func Foo(a A) {
}
```

根据结构体定义转换为函数 Foo 所需的类型变量

2.2 复合数据类型处理


jsValueToGo
goValueToJS
reflect.Ptr
Pointer of the Go data type, for example: *Struct or *string
reflect.Struct
The Go struct, for example: Struct, convert sub fields recursively
reflect.Slice
The Go data type array, for example []*Struct, []Struct, []string, []*string
Base data types
js.Value.Type
js.TypeObject
js.TypeUndefined

2.3 函数的移植方法


main
regInteropFunc
js.FuncOf(impl)
Impl
NewSheet
SetCellValue
…
regFuncs
global/window.objectName
regConstants
prepareArgs
jsValueToGo
goValueToJS

案例: Excelize 构建 WebAssembly 实践

excelize-wasm

A WebAssembly build of the Go Excelize library

excelize-wasm 是 Excelize 基础库的 WebAssembly / Javascript 实现，可用于操作 Office Excel 文档，基于 ECMA-376，ISO/IEC 29500 国际标准。可以使用它来读取、写入由 Excel、WPS、Number、Apache OpenOffice、Google Sheets 等办公应用创建的电子表格文档。支持 XLAM / XLSM / XLSX / XLTM / XLTX 等多种文档格式，高度兼容带有样式、图片(表)、透视表、切片器等复杂组件的文档。可应用于各类报表平台、云计算、边缘计算等系统。

Excelize 构建 WebAssembly 版本的目的，是为 JavaScript 和 TypeScript 语言调用提供支持，使得在浏览器和 Node.js 环境下使用 Excelize 处理电子表格办公文档成为可能。

2.4 编译优化

环境变量

```
CMD = ./cmd
DIST = ./dist
SRC = ./src
WASM = ${DIST}/excelize.wasm
```

单元测试

```
GOOS=js GOARCH=wasm go test -exec="${GOROOT}/misc/wasm/go_js_wasm_exec"
```

构建指令

```
GOOS=js GOARCH=wasm CGO_ENABLED=0 go build -v -a -ldflags="-w -s" \
    -gcflags=-trimpath=${GOPATH} \
    -asmflags=-trimpath=${GOPATH} \
    -o ../${WASM} main.go
# tinygo build -o ${WASM} -target wasm -no-debug main.go
gzip -f --best ${WASM} # go1.21.1 14MB -> 3.5MB | tinygo 0.29.0 3.9MB -> 1.4MB
```

目录布局

```
.
├── cmd
│   ├── go.mod
│   ├── go.sum
│   ├── main.go
│   └── main_test.go
├── dist
│   ├── LICENSE
│   ├── README.md
│   ├── excelize.wasm-
tinygo.gz
│   ├── index.d.ts
│   ├── index.js
│   ├── main.cjs
│   ├── main.js
│   └── package.json
├── node_modules
├── package.json
├── rollup.config.js
└── src
    ├── index.d.ts
    ├── index.js
    └── package.json
```

2.5 发布 WASM 文件

发布目录准备

```
cp LICENSE ${DIST}
cp README.md ${DIST}
cp ${SRC}/package.json ${DIST}
cp ${SRC}/index.d.ts ${DIST}
```

转译配置

https://github.com/xuri/excelize-wasm

选择分发版本：
IIFE
ESM
CJS

代码压缩：Terser

Google Developer Groups

启动 WASM

```
import $GOROOT/misc/wasm/wasm_exec.js

import pako from 'pako';

export async function init(wasmPath) {
    const go = new Go();
    var buffer;
    if (typeof window === 'undefined') {
        global.excelize = {};
        const fs = require('fs');
        buffer = pako.ungzip(fs.readFileSync(wasmPath));
    } else {
        window.excelize = {};
        buffer = pako.ungzip(await (await
fetch(wasmPath)).arrayBuffer());
    }
    if (buffer[0] === 0x1f && buffer[1] === 0x8b) {
        buffer = pako.ungzip(buffer);
    }
    const result = await WebAssembly.instantiate(buffer,
go.importObject);
    go.run(result.instance);
    return excelize;
};
```

类型提示

```
declare module 'excelize-wasm' {
    export function ColumnNumberToName(num: number):
        { col: string, error: string | null }
    // ...
    export enum CultureName {
        CultureNameUnknown,
        CultureNameEnUS,
        CultureNameZhCN,
    }
    export type Border = {
        Type?:  string;
        Color?: string;
        Style?: number;
    };
    export function init(path: string): Promise<{
        ColumnNumberToName: typeof ColumnNumberToName,
        CultureNameUnknown: typeof CultureName.CultureNameUnknown;
    }>;
}
```

发布类型声明文件

package.json

```
"exports": {
    "require": "./dist/main.cjs",
    "script": "./dist/index.js",
    "default": "./dist/main.js"
},
"files": [
    "excelize.wasm.gz",
    "index.d.ts",
    "index.js",
    "main.cjs",
    "main.js"
],
"types": "index.d.ts",
"typings": "index.d.ts”,
```

3.1 Node.js 运行 WebAssembly 案例

Go (go mod tidy)

```go
f := excelize.NewFile()
// 创建一个工作表
index := f.NewSheet("Sheet2")
// 设置单元格的值
f.SetCellValue("Sheet2", "A2", "Hello world.")
f.SetCellValue("Sheet1", "B2", 100)
// 获取工作表中指定单元格的值
cell, err := f.GetCellValue("Sheet1", "B2")
if err != nil {
    fmt.Println(err)
    return
}
fmt.Println(cell)
// 设置工作簿的默认工作表
f.SetActiveSheet(index)
// 根据指定路径保存文件
if err := f.SaveAs("Book1.xlsx"); err != nil {
    fmt.Println(err)
}
```

JavaScript (npm install excelize-wasm)

```javascript
const { init } = require('excelize-wasm');
const fs = require('fs');
init('./node_modules/excelize-wasm/
excelize.wasm.gz').then((excelize) => {
    const f = excelize.NewFile();
    // 创建一个工作表
    const { index } = f.NewSheet('Sheet2' );
    // 设置单元格的值
    f.SetCellValue('Sheet2', 'A2', 'Hello world.');
    f.SetCellValue('Sheet1', 'B2', 100);
    // 设置工作簿的默认工作表
    f.SetActiveSheet(index);
    // 根据指定路径保存文件
    const { buffer, error } = f.WriteToBuffer();
    if (error) { console.log(error); return; }
    fs.writeFile('Book1.xlsx', buffer, 'binary', (error) => {
        if (error) { console.log(error); }
    });
});
```

3.2 浏览器运行 WebAssembly 案例

```
<script src="https://<服务器地址>/excelize-wasm/index.js"></script>
<body>
    <div><button onclick="download()">下载</button></div>
    <script>
        function download() {
        excelizeWASM.init('https://<服务器地址>/excelize-wasm/excelize.wasm.gz').then((excelize) => {
            const f = excelize.NewFile();
            // 设置单元格的值
            f.SetCellValue('Sheet1', 'B2', 100);
            // 根据指定路径保存文件
            const { buffer, error } = f.WriteToBuffer();
            if (error) { console.log(error); return; }
            const link = document.createElement('a');
                link.download = 'Book1.xlsx';
                link.href = URL.createObjectURL(
                    new Blob([buffer], {
                    type: 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',
                })
            );
            link.click();
        });}
    </script>
</body>
```

Google Developer Groups

Beijing

Excelize: Go 语言 WebAssembly 移植实践

续日

Excelize Maintainer / Go Language Contributor

xuri.me@gmail.com

19 November, 2023

请备注 DevFest