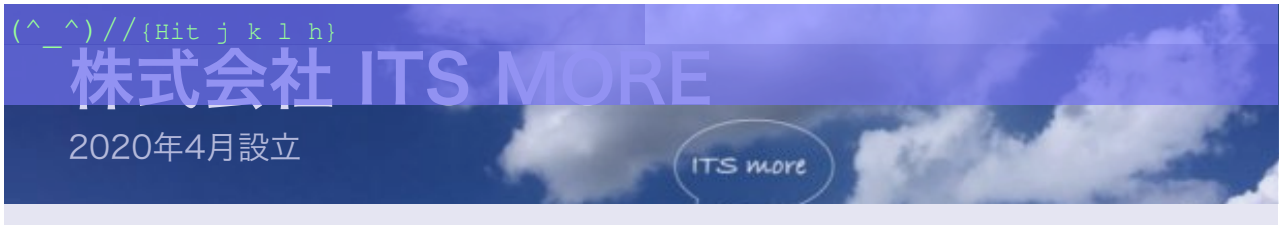


Digitally signed by 佐藤 豊
Date: 2020.09.21 04:00:22 +09'00'

2020年9月20日 投稿者: SATOXITS

GShell 0.4.7 – JavaScript/Golang通信

開発：昨日の件ですが、仕掛けておいたログから原因が明らかになりました。解決法も。

社長：果報は寝て待てですね。

基盤：寝たり起きたりしてますけどね。

暴走事案解決

開発：最大の原因というか90%の負荷は、DOMの表示に関する処理で、要するに textarea の表示高さ、.scrollHeight の計算。これはたぶん、ほぼゼロにできます。残り10%は JavaScriptの処理によるもので、要するに textarea の .value に対する文字列の追加。ここは、べたの textarea でなく構造化するかどうかという判断がからみます。textarea が1MBの状態に対して、追加するのに約20msかかっています。

社長：textareaはテキストのブラウズに便利なので活用したいですね。RAM to RAM の転写速度が10GB/s くらいとすると、10MB/ms くらいは出そうな気がしますが。

(^_^;; 開発：内部的にはtextareaの.valueのレベルでも、JavaScriptのstringのレベルでも構造化はされているのかなと思います。ブラウズだけなら1MB程度もサクサクスルスルで問題ないようです。ただ、.scrollHeightの計算だけが不自然に重い。固定ピッチフォントで画面幅が変わらないうちは、単純な差分の足し算とか、nowrapなら単純な掛け算で終わると思うのです。

社長：大きなバッファは動的に割り当てられる string じゃなくて、固定長のバイト列のバッファにしてしまうと良いのかもですね。

基盤：ターミナルのバッファとかは、1万行つまり10K行が普通ですね。1行50文字平均として、1MBで2万行はイケる計算です。

社長：まあ gsh.go.html も1万行がひとつのメドかなと思います。

基盤：すでに8000行を超えてますけどね。

開発：まあ現状、学習帳というか落書きノート状態ですから。整理すれば5000行くらいの内容かなと思います。

開発：あと、Safariでは「暴走」が起こらないように見えたのですが、勘違いで、textareaへのテキストの追加が累積していくとちゃんと重くなります。昨日、負荷原因として疑った、コンパイルされたコードの蓄積の問題は確認してませんが、これは今回の「暴走」にも見えた負荷の原因ではなかったようです。たただ、textareaへ追加と高さ再計算に関してSafariが軽いのは事実で、他のブラウザの2倍近く速いというか軽量です。独自のエンジンを持っているんだと思います。

GShell Golang部+JavaScript部の連携

開発：それで、GShellの応用例の一つとして、ホストマシンの負荷状況グラフをブラウザで描画したいと思います。

社長：会社設立直後にやってみようとしたことですが、ようやくたどり着きました。

(^_^) 開発：当時はこういう形で実現することになるとは想像しなかったですけどね。

開発：課題は、負荷情報の取得と、その描画の2つです。描画に関してはただDOMとJavaScriptを勉強するだけだと思います。問題は負荷情報の取得で、これをリアルタイムかつ軽量にやりたい。

社長：1秒に一度は解像度が低すぎますね。たとえば今回の暴走事件では、0.1秒単位では状況を観測したかった。そうすれば、もっと簡単に状況が理解できたはずです。

開発：で、問題は、技術的というよりセキュリティ上の制約から、JavaScriptでホストコンピュータのナマ情報を知ることが出来ないという点にあります。

社長：うちの場合、ユーザが見て良いって言ってるんですけどね。自分で作ったJavaScriptで自分のマシンを見たいだけです。JavaScriptの作者認証とか実行権限をちゃんと管理すれば問題無いと思うのですが。せっかく素晴らしい処理系があるのに、もったいない話です。

基盤：いっそ、外部には全く繋げないというブラウザかモードを作って、そこではJavaScriptに何でもやらせるってすれば良いんじゃないでしょうかね。ローカルなGUIプログラムのプラットフォームとして。

開発：まあブラウザはウェブ用、ネットワークアクセス用、JavaScript は作者不詳で全く信用しないっていう前提でしょうからね。

社長：なので、我社的な解決方法は、ローカルプログラムとして実行しているGolang GSellをサーバにして、ブラウザのGShell JavaScript部にローカル環境へのアクセスを与えるというものです。

基盤：Golang部からブラウザをopenして、接続ポートとか認証コードを引数で教えると良さそうに思います。

開発：たぶん、そうなりますね。

基盤：ローカルストレージ10MBというのは辛いし、ローカルマシンだけでし

(^_^) か共有できないのも残念なので、Golang部からNFSサービスとかもしてあげたら良いかと思います。

開発：それもやりたいですね。

社長：Golang部のGShellはネットワーク上に分散してバックエンドの世界を作る、JavaScript部はブラウザ内からそれにアクセスすることでどこからでもそこにアクセスできる。ブラウザによらない共通の世界。怪しげなHTTPサーバとかクラウドに頼らない。

開発：そうしたいですね。

WebSocketでコンニチハ

開発：それでまず、何で通信するかですが、前から目を付けていた WebSocket で行きたいと思います。接続後はただのソケットみたいに自由に使えると期待されますが、とりあえずHTTP要求・応答でも構いません。

基盤：例を見ると、めっちゃ簡単そうですね。

社長：Go側では実装されているんでしょうか？

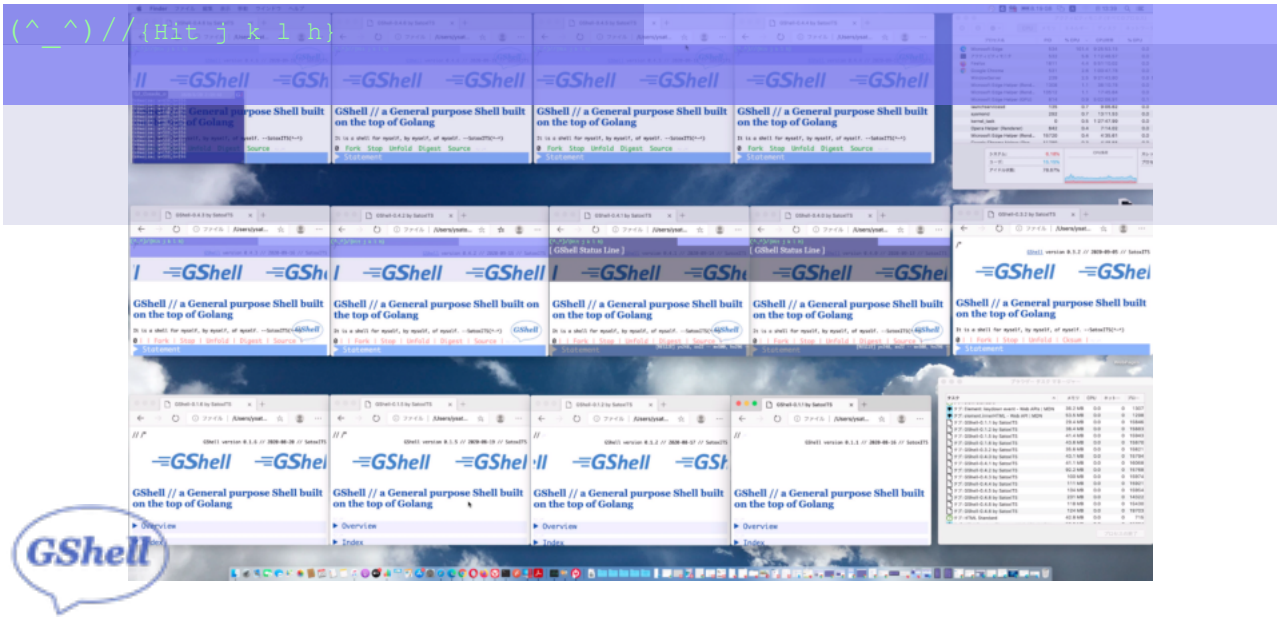
開発：websocket というパッケージがありますね。

基盤：例を見ると、めっちゃ簡単そうですね。

開発：なので、まず Hello応答をやりたいと思います。

基盤：というか、iMacのデスクトップがすごいことになってますが。

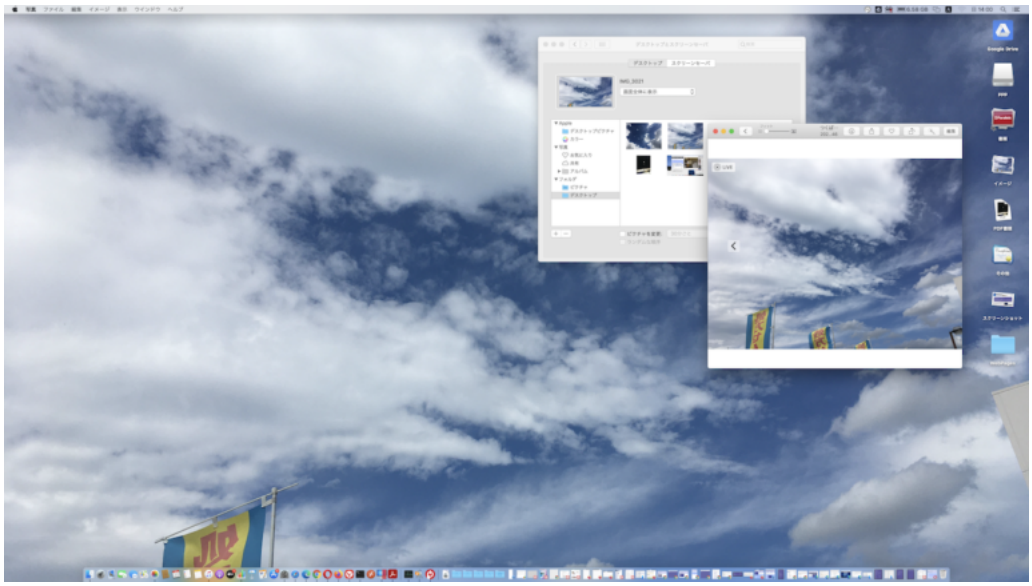
開発：昨日の問題を追跡した時の名残です。いずれ余裕ができたら整理しましょう。



開発：ミッションコントロールありがとう。

基盤：これ、仮想デスクトップごとに画面を変えたいですね。

社長：空の写真なら沢山あります。とりあえずこれかな。



開発：この旗がなんともいいですね。

社長：「処」だけでキレてるのが思わせぶりです。

基盤：うちのサイトの背景画像にしましょうか。

(^_^)/ こういえば

社長：そういえば昨日飲んだ帰りに、例の謎の定食屋のマスターにゥエルシァでばったり出会いました。ヒゲがーとか最近ご無沙汰とか言われちゃって。

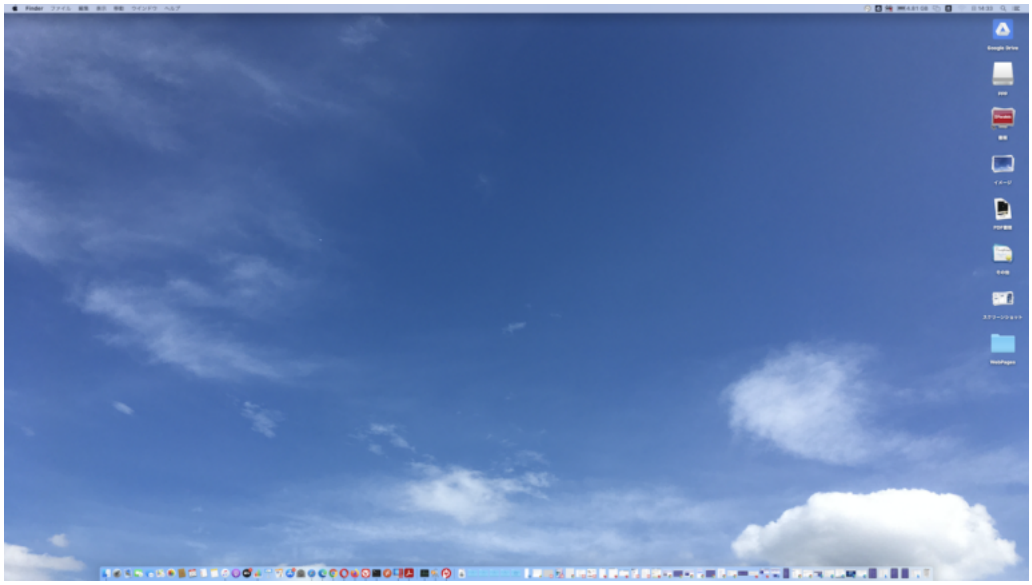
基盤：最近また、飲みに行くのを忘れる日が多いです。

社長：一度どこかで区切りをつけますかね。

開発：我が社初の慰安旅行！

基盤：そうだ、奈良へ行こう！

社長：あ、こういうのもいいかもですね。のどかで。



基盤：白い点、飛行機飛んでますね。成田行きでしょうか。

経理：秋の空も楽しみですね。

社長：夜空を撮れるカメラが欲しいです。

開発：さて、デスクトップも整いましたので。

基盤：BGMは吉田拓郎のままでいいんでしょうかw

社長：ちょっといっぷくしましょう。

(^_^)//{Hit j k l h}

Golang版 GShell WebSocket

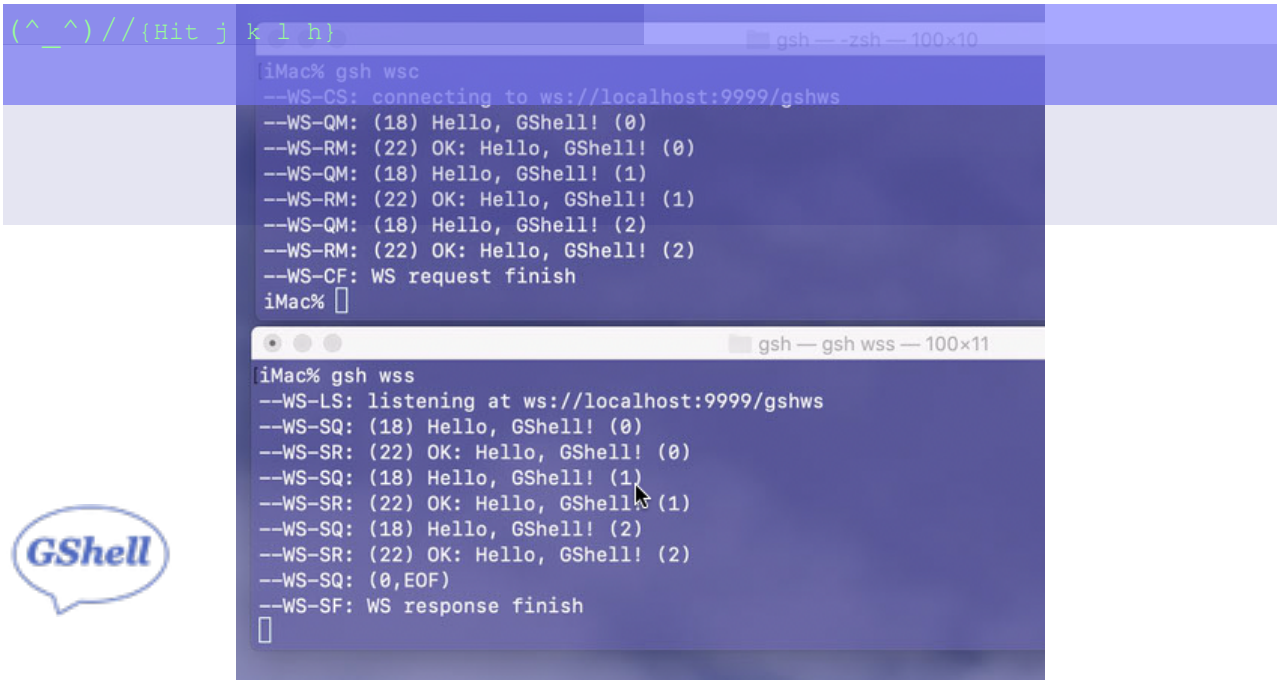
開発：まずGolang版のWebSocket、特に問題なく、テスト完了です。


```
// 2020-0920 WebSocket
// <a href="https://pkg.go.dev/golang.org/x/net/websocket">WS</a>
// <a href="https://godoc.org/golang.org/x/net/websocket">WS</a>
// INSTALL: go get golang.org/x/net/websocket
// import "golang.org/x/net/websocket"
const gshws_origin = "http://locahost:9999"
const gshws_port = "localhost:9999"
const gshws_path = "gshws"
const gshws_url = "ws://"+gshws_port+"/"+gshws_path
const GSHWS_MSGSIZE = (8*1024)
func fmtstring(fmts string, params ...interface{})(string){
        return fmt.Sprintf(fmts,params...)
}
func GSHWS_MARK(what string)(string){
        return "--WS-" + what + ": "
}
func gchk(what string,err error){
        if( err != nil ){
                panic(GSHWS_MARK(what)+err.Error())
        }
}
func glog(what string, fmts string, params ...interface{}){
        fmt.Print(GSHWS_MARK(what))
        fmt.Printf(fmts+"\n",params...)
}

func serv1(ws *websocket.Conn) {
        var reqb = make([]byte,GSHWS_MSGSIZE)
        for {
                rn, rerr := ws.Read(reqb)
                if( rerr != nil || rn < 0 ){
                        glog("SQ",fmtstring("(%v,%v)",rn,rerr))
                        break
                }
                req := string(reqb[0:rn])
                glog("SQ",fmtstring("(%v) %v",rn,req))
                res := fmtstring("OK: %v",req)
                wn, werr := ws.Write([]byte(res))
                gchk("SE",werr)
                glog("SR",fmtstring("(%v) %v",wn,string(res)))
        }
        glog("SF","WS response finish")
}
func ws_server(argv []string) {
        port := gshws_port
        glog("LS",fmtstring("listening at %v",gshws_url))
        http.Handle("/"+gshws_path,websocket.Handler(serv1))
        err := http.ListenAndServe(port,nil)
        gchk("LE",err)
}

func ws_client(argv []string) {
        glog("CS",fmtstring("connecting to %v",gshws_url))
        ws, err := websocket.Dial(gshws_url,"",gshws_origin)
        gchk("C",err)

        var resb = make([]byte, GSHWS_MSGSIZE)
        for qi := 0; qi < 3; qi++ {
                req := fmtstring("Hello, GShell! (%v)",qi)
                wn, werr := ws.Write([]byte(req))
                glog("QM",fmtstring("(%v) %v",wn,req))
                gchk("QE",werr)
                rn, rerr := ws.Read(resb)
                gchk("RE",rerr)
                glog("RM",fmtstring("(%v) %v",rn,string(resb)))
        }
        glog("CF","WS request finish")
}
```



開発：実は驚いたことに、すでにGolangの事をかなり忘れていて、最初は間違ってwriteでバッファ全体を送ってしまったのですが、それでも繰り返しの通信が出来ました。

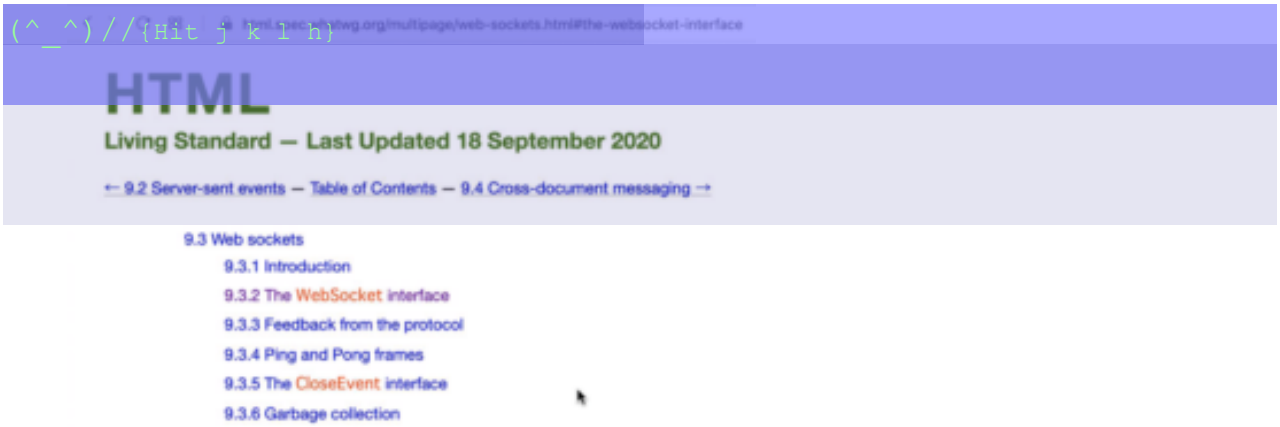
社長：つまり、デフォルトでは無限オクテットストリームでは無いってことですかね。

基盤：デフォルトのポート番号は 9999 にするのでしょうか？

開発：仮決めですが、そうなる可能性が高いです。正式に登録されてなくて4桁で、昔からずっと使ってきた番号です。

JavaScript版 GShell WebSocket

開発：でいよいよお待ちかねの JavaScript版です。MDNから指されている仕様は、これ。



基盤：更新日付がこの9/18ですね。とってもライブ感。

開発：まあ、WebSocketに関して変更があったのかどうかはわかりませんけど。

社長：JavaScriptの場合には、サーバにはなれないんでしょうか？

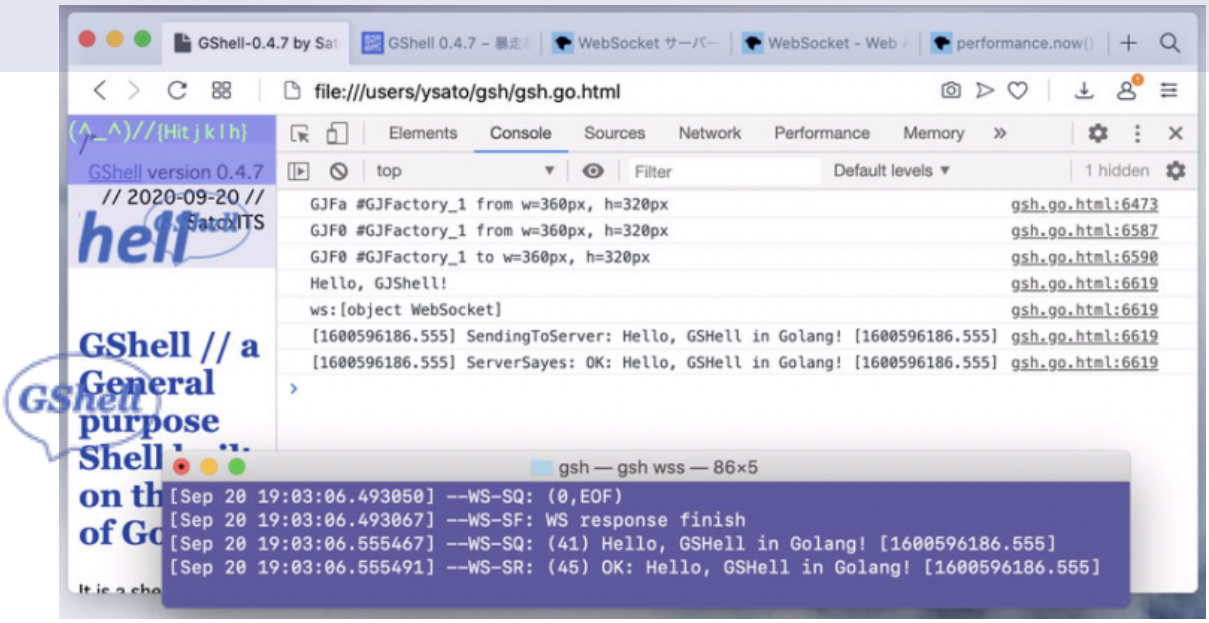
開発：少なくともこの仕様はクライアント側のみですね。Node.js でなら、サーバ側を作る例があちこちにありますが。ただ要するに WebSocketは、まず HTTPサーバで受信して、コネクションを Upgrade する形でできているようですから、HTTPサーバ経由でならサーバになれるのではないかと思われます。

基盤：その websocket 上の ping って最優先みたいですから、サーバの負荷の影響が少なくて、ネットワーク状況を知るのに使えると良いですね。

開発：MDNの例によると、とりあえずコードはこれだけで良いはずです。

```
<script>
var ws = new WebSocket("ws://localhost:9999/gshws");
console.log('ws:'+ws);
ws.addEventListener('open', function(event){
        now = new Date().getTime() / 1000;
        msg = 'Hello, GSHell in Golang! ['+now+']';
        console.log('['+now+'] SendingToServer: '+msg);
        ws.send(msg);
});
ws.addEventListener('message', function(event){
        now = new Date().getTime() / 1000;
        console.log('['+now+'] ServerSayes: '+event.data);
});
</script>
```

開発：こちらでGolang版GShellサーバを立ち上げておいて、ブラウザでGShellのHTMLをリロード。



基盤：パチパチパチ。

社長：めでたし。祝杯を挙げに行きましょう。とっても giant leap。

開発：応答は1ミリ秒程度のようです。

基盤：JavaScript から Golang の GShell IME を使えそうですね。

開発：問題なく。

基盤：syscall も全部呼べる。

開発：問題なく。

社長：GShellサーバがリモートにあってもOK。

開発：問題なく。

基盤：ローカルファイルも使い放題。

{^_^} 開発：問題なく。

基盤：とりあえず GJ Console から uptime コマンドを打ちたいです。top でも良いです。

開発：OSのコマンドを投げて、Goで実行した結果を返せばなんでもできますね。

社長：ようやく、Golang部とJavaScript部がつながりました。

開発：もっと早くにこれをやれば良かったですかね。

社長：それはどうですかね。ともかくめでたいので飲みに行きましょう。今日は謎の定食屋かな。

— 2020-0920 SatoxITS

gsh-0.4.7.go_ ダウンロード

社長：ただい…ま…く。苦しい…

経理：大丈夫ですか？

社長：食べすぎました…

基盤：謎の定食屋で食べ放題＋飲み放題？

社長：あー…あそこは今日はマスターがやる気が無かったのか閉まってまして、さたぽんで。先日のツケもありましたし…

＊　＊　＊

社長：ふぅ。少し落ち着きました。

開発：胃の中で食べたものの膨張が収まったですかね。

社長：それで、お店のWiFiでGoogleのニュースをブラウズしてたのですが、いくつか新情報。その1、ただいまUS女子ツアー開催中、明日というか日本時間で今日深夜スタートです。

開発：リーダーズボード開けときます。また寝ちゃう気もしますけど。

基盤：開催地はうちの支部のあるオレゴンですね。

社長：その2、藤井2冠は自作PCで将棋研究をしており、そのCPUは50万円。ライゼンスレッドリッパー3990Xとかいうもの。

基盤：それ、アマゾンで完成品が50万円で売ってますね。2.9GHzですが、64コアというのが凄いですかね。それでも消費電力280W程度の模様。

社長：あとは、ポケモンGOの記事が、Goの誤植に見える件とか。世間は4連休中であるらしいとか。

社長：ああそれで考えたんですが、これはやはりJavaScript側からGolang側に指示するより、Golang側が主導で、WebSocket経由でJavaScriptに描画指示する形のほうが自然なのかなと。性能的にも、セキュリティ的にも。接続は現状、JavaScript側からするという選択になるとしても。

開発：まあブラウザをWindowサーバとして使いたい、JavaScirpt をそのための処理系として使う、というのはもともとの考え方でした。でもこれは、どちらが主導するのが適切かは、アプリによると思いますが。

社長：まあそうでしょうね。いずれにしてもそれを念頭に置いて、明日以降を進めたいと思います。

/* */ /*

GShell version 0.4.7 // 2020-09-20 // SatoxITS

(^_^)//{Hit j k l h}

GShell // a General purpose Shell built on the top of Golang

It is a shell for myself, by myself, of myself. -SatoxITS(^-^)

0 Fork Stop Unfold Digest Source */ /*

- Statement

/ /

- Features

/ /

- Index

*/ //

- Go Source

//

- Considerations

// /*

- References

/ /

- Raw Source

/ /

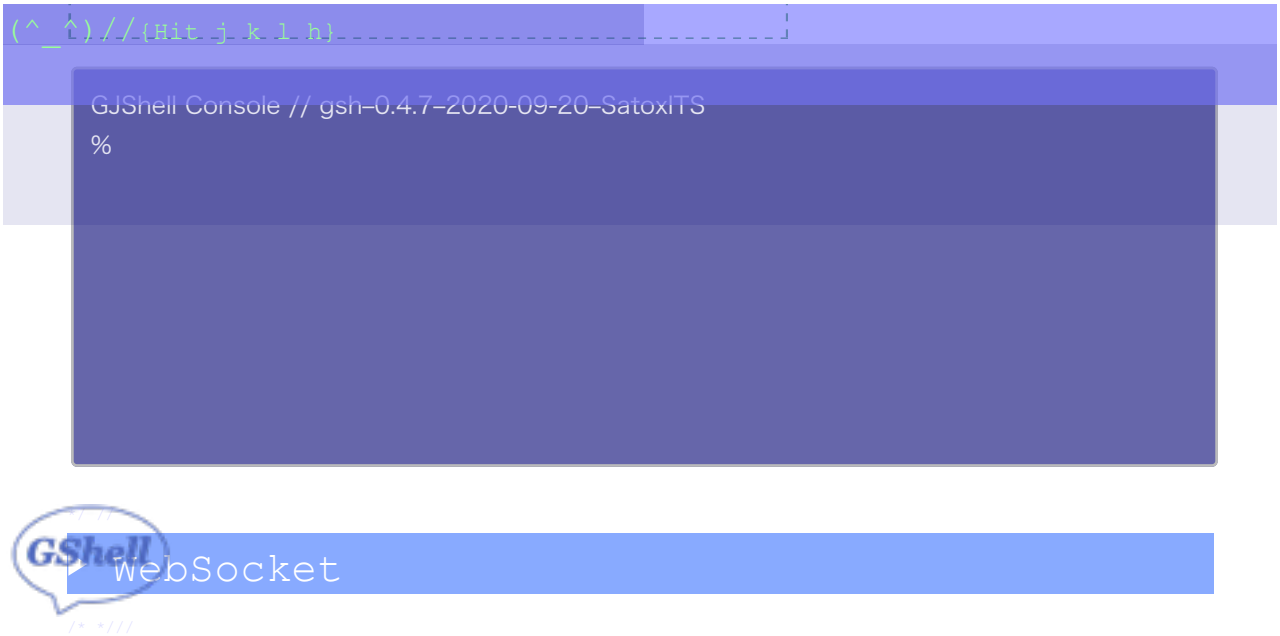
- GJScript

/ /

GJ Factory_1

GJ_Console_0 2020/8/21 GJ

```
-- GShell: gsh-0.4.7-2020-09-20-SatoxITS
-- Digest: 2443459636 211864
-- Display: screen=2560px, window=1047px
OnScroll: x=64,y=10680
OnScroll: x=64,y=10679
OnScroll: x=64,y=10678
OnScroll: x=64,y=10676
OnScroll: x=64,y=10674
OnScroll: x=64,y=10672
OnScroll: x=64,y=10668
OnScroll: x=64,y=10664
OnScroll: x=64,y=10659
OnScroll: x=64,y=10653
OnScroll: x=64,y=10647
OnScroll: x=64,y=10641
[---]
```


(^_^)//{Hit j k l h}
GJShell Console // gsh-0.4.7-2020-09-20-SatoxITS
%
GShell
WebSocket