Javascript to Go Cheat Sheet

Javascript	**Go**

somefunction()	func main() {
	  somefunction()
	}

var myFunc = function() {};	var myFunc = func(){}

function myFunc() {
 	
}
	func myFunc() {
	
	}

var x = 5;	var x int = 5 (anywhere) x := 5 (only in func)
var x = 5; x = “Hello”;	// can’t do this
// no constants	const x = 5
var x = 1, y = 2;	var x, y = 1, 2
var x; // undefined	var x int // 0
“Hello World”	var mySaying string = “Hello” var myBacktick string = `He“ll”o`
1234	1234
1234.2	1234.2

+, -, /, %	+, -, /, %
true, false && ||	true, false && ||
x === y	x == y

if (i < 10)
{
 	
}
else if (i < 20)
{
 	
}
else
{
 	
}
	if i < 10 {
	} else if (i < 20) {
	} else {
	
	}

while (whatever) {
 	
}
	for whatever {
	
	}

while (true) {
 	
}
	for {
	
	}

for (var i = 0; i < 10; i++) {
 	
}
	for i := 0; i < 10; i++ {
	
	}

var i;
for (i = 0; i < 10; i++) {
 	
}
	var i int
	for i = 0; i < 10; i++ {
	
	}

var obj = {
“x”: “y”,
“y”: 10,
};
for (var key in obj) {
console.log(“Key is:”, key);
console.log(“Value is:”, obj[key]);
}
	obj := map[string]string {
	"x": "y",
	"y": "z",
	}
	for key := range obj {
	fmt.Println(“Key is:”, key)
	fmt.Println(“Value is:”, obj[key])
	}

var xs = [1,2,3,4];	xs := [4]int {1,2,3,4}
var xs = [1,2,3,4]; xs.push(5,6,7,8);	xs := []int {1,2,3,4} xs = append(xs,5,6,7,8)

// add to head
<script src=”fmt.js”></script>
// fmt.js
fmt = {
Println: function() { }
}
// or
var fmt = require(“fmt”)
	import “fmt”
	
	// fmt.go
	func Println() {
	
	}

function sum() {
for (var i=0; i<arguments.length;
i++) {
 	
}
	func sum(xs ...int) int {
	for key, value := range xs {
	
	}
	} sum(1,2,3) sum([]{1,2,3}

}
 	
(function(n) {
    if (n == 0 || n == 1) {
       return 1;
    } else {
       return n *
arguments.callee(n-1);
    }
})(5)
	var factorial func(int) int
	factorial = func(n int) int {
	    if n == 0 || n == 1 {
	        return 1
	    } else {
	        return n * factorial(n-1)
	    }
	}

function MyClass(x) {
  this.x = x;
}
MyClass.prototype = {
  “whatever”: function() {
     console.log(this.x);
  }
};
MyClass.prototype.someOtherMethod =
function() {
 	
};
var obj = new MyClass(5);
obj.whatever();
	type MyClass struct {
	  x int
	}
	func NewMyClass(x int) *MyClass {
	  return &MyClass{
	    x: x,
	  }
	}
	func (this *MyClass) whatever() {
	  fmt.Println(this.x)
	} bs, err := ioutil.ReadAll(f)
	     if err != nil {
	          log.Fatalln("my program broke")
	     }
	
	     str := string(bs)
	func main() {
	  obj := NewMyClass(5)
	  obj.whatever()
	}

var str = JSON.stringify({ “a”: “b”})	bs, err : = json.Marshal(map[string]string{“a”:”b”})

try {
var obj = JSON.parse(str)
} catch(err) {
 	
}
	var obj map[string]string
	err := json.Unmarshal(str, &obj)