





[Discover Packages](#) > [Standard library](#) > [syscall](#) > [js](#) 

js

package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: 3 |Imported by: [1,382](#)**Details** Valid [go.mod](#) file   Redistributable license   Tagged version  Stable version [Learn more](#)**Repository**[cs.opensource.google/go/go](#)**Links** [Report a Vulnerability](#) Documentation <> **Documentation**[Rendered for js/wasm](#)**Overview**

Package js gives access to the WebAssembly host environment when using the js/wasm architecture. Its API is based on JavaScript semantics.

This package is EXPERIMENTAL. Its current scope is only to allow tests to run, but not yet to provide a comprehensive API for users. It is exempt from the Go compatibility promise.

Index[func CopyBytesToGo\(dst \[\]byte, src Value\) int](#)[func CopyBytesToJS\(dst Value, src \[\]byte\) int](#)[type Error](#)[func \(e Error\) Error\(\) string](#)[type Func](#)[func FuncOf\(fn func\(this Value, args \[\]Value\) any\) Func](#)[func \(c Func\) Release\(\)](#)[type Type](#)[func \(t Type\) String\(\) string](#)[type Value](#)[func Global\(\) Value](#)[func Null\(\) Value](#)[func Undefined\(\) Value](#)[func ValueOf\(x any\) Value](#)[func \(v Value\) Bool\(\) bool](#)

func (v Value) Call(m string, args ...any) Value

func (v Value) Delete(p string)

func (v Value) Equal(w Value) bool

func (v Value) Float() float64

func (v Value) Get(p string) Value

func (v Value) Index(i int) Value

func (v Value) InstanceOf(t Value) bool

func (v Value) Int() int

func (v Value) Invoke(args ...any) Value

func (v Value) IsNaN() bool

func (v Value) IsNull() bool

func (v Value) IsUndefined() bool

func (v Value) Length() int

func (v Value) New(args ...any) Value

func (v Value) Set(p string, x any)

func (v Value) SetIndex(i int, x any)

func (v Value) String() string

func (v Value) Truthy() bool

func (v Value) Type() Type

type ValueError

func (e *ValueError) Error() string

Examples

FuncOf

Constants

This section is empty.

Variables

This section is empty.

Functions

func CopyBytesToGo

added in go1.13

```
func CopyBytesToGo(dst []byte, src Value) int
```

CopyBytesToGo copies bytes from src to dst. It panics if src is not an Uint8Array or Uint8ClampedArray. It returns the number of bytes copied, which will be the minimum of the lengths of src and dst.

func CopyBytesToJS

added in go1.13

```
func CopyBytesToJS(dst Value, src []byte) int
```

CopyBytesToJS copies bytes from src to dst. It panics if dst is not an Uint8Array or Uint8ClampedArray. It returns the number of bytes copied, which will be the minimum of the lengths of src and dst.

Types

type **Error**

```
type Error struct {  
    // Value is the underlying JavaScript error value.  
    Value  
}
```

Error wraps a JavaScript error.

func (Error) **Error**

```
func (e Error) Error() string
```

Error implements the error interface.

type **Func**

added in go1.12

```
type Func struct {  
    Value // the JavaScript function that invokes the Go function  
    // contains filtered or unexported fields  
}
```

Func is a wrapped Go function to be called by JavaScript.

func **FuncOf**

added in go1.12

```
func FuncOf(fn func(this Value, args []Value) any) Func
```

FuncOf returns a function to be used by JavaScript.

The Go function fn is called with the value of JavaScript's "this" keyword and the arguments of the invocation. The return value of the invocation is the result of the Go function mapped back to JavaScript according to ValueOf.

Invoking the wrapped Go function from JavaScript will pause the event loop and spawn a new goroutine. Other wrapped functions which are triggered during a call from Go to JavaScript get executed on the same goroutine.

As a consequence, if one wrapped function blocks, JavaScript's event loop is blocked until that function returns. Hence, calling any async JavaScript API, which requires the event loop, like fetch (http.Client), will cause an immediate deadlock. Therefore a blocking function should explicitly start a new goroutine.

Func.Release must be called to free up resources when the function will not be invoked any more.

► [Example](#)

func (Func) Release

added in go1.12

```
func (c Func) Release()
```

Release frees up resources allocated for the function. The function must not be invoked after calling Release. It is allowed to call Release while the function is still running.

type Type

```
type Type int
```

Type represents the JavaScript type of a Value.

```
const (  
    TypeUndefined Type = iota  
    TypeNull  
    TypeBoolean  
    TypeNumber  
   .TypeString  
   .TypeSymbol  
    TypeObject  
    TypeFunction  
)
```

func (Type) String

```
func (t Type) String() string
```

type Value

```
type Value struct {  
    // contains filtered or unexported fields  
}
```

Value represents a JavaScript value. The zero value is the JavaScript value "undefined". Values can be checked for equality with the Equal method.

func Global

```
func Global() Value
```

Global returns the JavaScript global object, usually "window" or "global".

func Null

```
func Null() Value
```

Null returns the JavaScript value "null".

func Undefined

```
func Undefined() Value
```

Undefined returns the JavaScript value "undefined".

func ValueOf

```
func ValueOf(x any) Value
```

ValueOf returns x as a JavaScript value:

Go	JavaScript	
-----	-----	
js.Value	[its value]	
js.Func	function	
nil	null	
bool	boolean	
integers and floats	number	
string	string	
[]interface{}	new array	
map[string]interface{}	new object	

Panics if x is not one of the expected types.

func (Value) Bool

```
func (v Value) Bool() bool
```

Bool returns the value v as a bool. It panics if v is not a JavaScript boolean.

func (Value) Call

```
func (v Value) Call(m string, args ...any) Value
```

Call does a JavaScript call to the method m of value v with the given arguments. It panics if v has no method m. The arguments get mapped to JavaScript values according to the ValueOf function.

func (Value) Delete

added in go1.14

```
func (v Value) Delete(p string)
```

Delete deletes the JavaScript property p of value v. It panics if v is not a JavaScript object.

func (Value) Equal

added in go1.14

```
func (v Value) Equal(w Value) bool
```

Equal reports whether v and w are equal according to JavaScript's === operator.

func (Value) Float

```
func (v Value) Float() float64
```

Float returns the value v as a float64. It panics if v is not a JavaScript number.

func (Value) Get

```
func (v Value) Get(p string) Value
```

Get returns the JavaScript property p of value v. It panics if v is not a JavaScript object.

func (Value) Index

```
func (v Value) Index(i int) Value
```

Index returns JavaScript index i of value v. It panics if v is not a JavaScript object.

func (Value) InstanceOf

```
func (v Value) InstanceOf(t Value) bool
```

InstanceOf reports whether v is an instance of type t according to JavaScript's instanceof operator.

func (Value) Int

```
func (v Value) Int() int
```

Int returns the value v truncated to an int. It panics if v is not a JavaScript number.

func (Value) Invoke

```
func (v Value) Invoke(args ...any) Value
```

Invoke does a JavaScript call of the value v with the given arguments. It panics if v is not a JavaScript function. The arguments get mapped to JavaScript values according to the ValueOf function.

func (Value) IsNaN

added in go1.14

```
func (v Value) IsNaN() bool
```

IsNaN reports whether v is the JavaScript value "NaN".

func (Value) IsNull

added in go1.14

```
func (v Value) IsNull() bool
```

IsNull reports whether v is the JavaScript value "null".

func (Value) IsUndefined

added in go1.14

```
func (v Value) IsUndefined() bool
```

IsUndefined reports whether v is the JavaScript value "undefined".

func (Value) Length

```
func (v Value) Length() int
```

Length returns the JavaScript property "length" of v. It panics if v is not a JavaScript object.

func (Value) New

```
func (v Value) New(args ...any) Value
```

New uses JavaScript's "new" operator with value v as constructor and the given arguments. It panics if v is not a JavaScript function. The arguments get mapped to JavaScript values according to the ValueOf function.

func (Value) Set

```
func (v Value) Set(p string, x any)
```

Set sets the JavaScript property p of value v to ValueOf(x). It panics if v is not a JavaScript object.

func (Value) SetIndex

```
func (v Value) SetIndex(i int, x any)
```

SetIndex sets the JavaScript index i of value v to ValueOf(x). It panics if v is not a JavaScript object.

func (Value) String

```
func (v Value) String() string
```

String returns the value v as a string. String is a special case because of Go's String method convention. Unlike the other getters, it does not panic if v's Type is not TypeString. Instead, it returns a string of the form "<T>" or "<T: V>" where T is v's type and V is a string representation of v's value.

func (Value) Truthy

added in go1.12

```
func (v Value) Truthy() bool
```

Truthy returns the JavaScript "truthiness" of the value v. In JavaScript, false, 0, "", null, undefined, and NaN are "falsy", and everything else is "truthy". See <https://developer.mozilla.org/en-US/docs/Glossary/Truthy>.

func (Value) Type

```
func (v Value) Type() Type
```

Type returns the JavaScript type of the value v. It is similar to JavaScript's typeof operator, except that it returns TypeNull instead of TypeObject for null.

type ValueError

```
type ValueError struct {  
    Method string  
    Type   Type  
}
```

A ValueError occurs when a Value method is invoked on a Value that does not support it. Such cases are documented in the description of each method.

func (*ValueError) Error

```
func (e *ValueError) Error() string
```



Source Files

[View all](#) 

[func.go](#)

[js.go](#)

Why Go

[Use Cases](#)

[Case Studies](#)

Get Started

[Playground](#)

[Tour](#)

[Stack Overflow](#)

[Help](#)

Packages

[Standard Library](#)

[About Go Packages](#)

About

[Download](#)

[Blog](#)

[Issue Tracker](#)

[Release Notes](#)

[Brand Guidelines](#)

[Code of Conduct](#)

Connect

[Twitter](#)

[GitHub](#)

[Slack](#)

[r/golang](#)

[Meetup](#)

[Golang Weekly](#)

[Copyright](#)

[Terms of Service](#)

[Privacy Policy](#)

[Report an Issue](#)



[Google](#)