-

https://pkg.go.dev/github.com/gogf/gf/v2/encoding/gjson

New

- NewdataJsondatamapslice
- safeJsonfalse
-

```
func New(data interface{}, safe ...bool) *Json
```

-

```
func ExampleNew() {
        jsonContent := `{"name":"john", "score":"100"}`
        j := gjson.New(jsonContent)
        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))

        // Output:
        // john
        // 100
}
```

NewWithTag

- NewWithTagdataJsondatamapslice
- tgtsmap','
- safeJsonfalse
-

```
func NewWithTag(data interface{}, tags string, safe ...bool) *Json
```

-

```
func ExampleNewWithTag() {
        type Me struct {
                Name  string `tag:"name"`
                Score int    `tag:"score"`
                Title string
        }
        me := Me{
                Name:  "john",
                Score: 100,
                Title: "engineer",
        }
        j := gjson.NewWithTag(me, "tag", true)
        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))
        fmt.Println(j.Get("Title"))

        // Output:
        // john
        // 100
        // engineer
}
```

NewWithOptions

Content Menu

- NewWithOptionsdataJsondatamapslice
-

```
func NewWithOptions(data interface{}, options Options) *Json
```

-

```
func ExampleNewWithOptions() {
        type Me struct {
                Name  string `tag:"name"`
                Score int    `tag:"score"`
                Title string
        }
        me := Me{
                Name:  "john",
                Score: 100,
                Title: "engineer",
        }

        j := gjson.NewWithOptions(me, gjson.Options{
                Tags: "tag",
        })
        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))
        fmt.Println(j.Get("Title"))

        // Output:
        // john
        // 100
        // engineer
}
```

```
func ExampleNewWithOptions_UTF8BOM() {
        jsonContent := `{"name":"john", "score":"100"}`

        content := make([]byte, 3, len(jsonContent)+3)
        content[0] = 0xEF
        content[1] = 0xBB
        content[2] = 0xBF
        content = append(content, jsonContent...)

        j := gjson.NewWithOptions(content, gjson.Options{
                Tags: "tag",
        })
        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))

        // Output:
        // john
        // 100
}
```

Load

- LoadpathJson
-

```
func Load(path string, safe ...bool) (*Json, error)
```

-

```
func ExampleLoad() {
        jsonFilePath := gtest.DataPath("json", "data1.json")
        j, _ := gjson.Load(jsonFilePath)
        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))

        notExistFilePath := gtest.DataPath("json", "data2.json")
        j2, _ := gjson.Load(notExistFilePath)
        fmt.Println(j2.Get("name"))

        // Output:
        // john
        // 100
}
```

```
func ExampleLoad_Xml() {
        jsonFilePath := gtest.DataPath("xml", "data1.xml")
        j, _ := gjson.Load(jsonFilePath)
        fmt.Println(j.Get("doc.name"))
        fmt.Println(j.Get("doc.score"))
}
```

LoadJson

- LoadJsonJSONJson
-

```
func LoadJson(data interface{}, safe ...bool) (*Json, error)
```

-

```
func ExampleLoadJson() {
        jsonContent := `{"name":"john", "score":"100"}`
        j, _ := gjson.LoadJson(jsonContent)
        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))

        // Output:
        // john
        // 100
}
```

LoadXml

- LoadXmlXMLJson
-

```
func LoadXml(data interface{}, safe ...bool) (*Json, error)
```

-

```
func ExampleLoadXml() {
        xmlContent := `<?xml version="1.0" encoding="UTF-8"?>
        <base>
                <name>john</name>
                <score>100</score>
        </base>`
        j, _ := gjson.LoadXml(xmlContent)
        fmt.Println(j.Get("base.name"))
        fmt.Println(j.Get("base.score"))

        // Output:
        // john
        // 100
}
```

LoadIni

- LoadIniINIJson
-

```
func LoadIni(data interface{}, safe ...bool) (*Json, error)
```

-

```
func ExampleLoadIni() {
        iniContent := `
        [base]
        name = john
        score = 100
        `
        j, _ := gjson.LoadIni(iniContent)
        fmt.Println(j.Get("base.name"))
        fmt.Println(j.Get("base.score"))

        // Output:
        // john
        // 100
}
```

LoadYaml

- LoadYamlYAMLJson
-

```
func LoadYaml(data interface{}, safe ...bool) (*Json, error)
```

-

```
func ExampleLoadYaml() {
        yamlContent :=
                `base:
  name: john
  score: 100`

        j, _ := gjson.LoadYaml(yamlContent)
        fmt.Println(j.Get("base.name"))
        fmt.Println(j.Get("base.score"))

        // Output:
        // john
        // 100
}
```

LoadToml

- LoadTomlTOMLJson
-

```
func LoadToml(data interface{}, safe ...bool) (*Json, error)
```

-

```
func ExampleLoadToml() {
        tomlContent :=
                `[base]
  name = "john"
  score = 100`

        j, _ := gjson.LoadToml(tomlContent)
        fmt.Println(j.Get("base.name"))
        fmt.Println(j.Get("base.score"))

        // Output:
        // john
        // 100
}
```

LoadContent

- LoadContentJsoncontent:JSON, XML, INI, YAMLTOML
-

```
func LoadContent(data interface{}, safe ...bool) (*Json, error)
```

-

```
func ExampleLoadContent() {
        jsonContent := `{"name":"john", "score":"100"}`

        j, _ := gjson.LoadContent(jsonContent)

        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))

        // Output:
        // john
        // 100
}
```

```
func ExampleLoadContent_UTF8BOM() {
        jsonContent := `{"name":"john", "score":"100"}`

        content := make([]byte, 3, len(jsonContent)+3)
        content[0] = 0xEF
        content[1] = 0xBB
        content[2] = 0xBF
        content = append(content, jsonContent...)

        j, _ := gjson.LoadContent(content)

        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))

        // Output:
        // john
        // 100
}
```

```
func ExampleLoadContent_Xml() {
        xmlContent := `<?xml version="1.0" encoding="UTF-8"?>
        <base>
                <name>john</name>
                <score>100</score>
        </base>`

        x, _ := gjson.LoadContent(xmlContent)

        fmt.Println(x.Get("base.name"))
        fmt.Println(x.Get("base.score"))

        // Output:
        // john
        // 100
}
```

LoadContentType

- LoadContentTypeJson:Json, XML, INI, YAMLTOML
-

```
func LoadContentType(dataType string, data interface{}, safe ...
bool) (*Json, error)
```

-

```
func ExampleLoadContentType() {
        jsonContent := `{"name":"john", "score":"100"}`
        xmlContent := `<?xml version="1.0" encoding="UTF-8"?>
        <base>
                <name>john</name>
                <score>100</score>
        </base>`

        j, _ := gjson.LoadContentType("json", jsonContent)
        x, _ := gjson.LoadContentType("xml", xmlContent)
        j1, _ := gjson.LoadContentType("json", "")

        fmt.Println(j.Get("name"))
        fmt.Println(j.Get("score"))
        fmt.Println(x.Get("base.name"))
        fmt.Println(x.Get("base.score"))
        fmt.Println(j1.Get(""))

        // Output:
        // john
        // 100
        // john
        // 100
}
```

IsValidDataType

- `IsValidDataTypedataType`
-

```
func IsValidDataType(dataType string) bool
```

-

```
func ExampleIsValidDataType() {
        fmt.Println(gjson.IsValidDataType("json"))
        fmt.Println(gjson.IsValidDataType("yml"))
        fmt.Println(gjson.IsValidDataType("js"))
        fmt.Println(gjson.IsValidDataType("mp4"))
        fmt.Println(gjson.IsValidDataType("xsl"))
        fmt.Println(gjson.IsValidDataType("txt"))
        fmt.Println(gjson.IsValidDataType(""))
        fmt.Println(gjson.IsValidDataType(".json"))

        // Output:
        // true
        // true
        // true
        // false
        // false
        // false
        // false
        // true
}
```

Valid

- `ValiddataJSON dataJSONbytesstring`
-

```
func Valid(data interface{}) bool
```

-

```
func ExampleValid() {
        data1 := []byte(`{"n":123456789, "m":{"k":"v"}, "a":[1,2,3]}
`)
        data2 := []byte(`{"n":123456789, "m":{"k":"v"}, "a":[1,2,3]`)
        fmt.Println(gjson.Valid(data1))
        fmt.Println(gjson.Valid(data2))

        // Output:
        // true
        // false
}
```

Marshal

- MarshalEncode
-

```
func Marshal(v interface{}) (marshaledBytes []byte, err error)
```

-

```
func ExampleMarshal() {
        data := map[string]interface{}{
                "name":  "john",
                "score": 100,
        }

        jsonData, _ := gjson.Marshal(data)
        fmt.Println(string(jsonData))

        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "Guo Qiang",
                Age:  18,
        }

        infoData, _ := gjson.Marshal(info)
        fmt.Println(string(infoData))

        // Output:
        // {"name":"john","score":100}
        // {"Name":"Guo Qiang","Age":18}
}
```

MarshalIndent

- MarshalIndentjson.MarshalIndent
-

```
func MarshalIndent(v interface{}, prefix, indent string)
(marshaledBytes []byte, err error)
```

-

```
func ExampleMarshalIndent() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        infoData, _ := gjson.MarshalIndent(info, "", "\t")
        fmt.Println(string(infoData))

        // Output:
        // {
        //         "Name": "John",
        //         "Age": 18
        // }
}
```

Unmarshal

- UnmarshalDecodeTo
-

```
func Unmarshal(data []byte, v interface{}) (err error)
```

-

```
func ExampleUnmarshal() {
        type BaseInfo struct {
                Name  string
                Score int
        }

        var info BaseInfo

        jsonContent := "{\"name\":\"john\",\"score\":100}"
        gjson.Unmarshal([]byte(jsonContent), &info)
        fmt.Printf("%+v", info)

        // Output:
        // {Name:john Score:100}
}
```

Encode

- EncodevalueJSONbyte
-

```
func Encode(value interface{}) ([]byte, error)
```

-

```
func ExampleEncode() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        infoData, _ := gjson.Encode(info)
        fmt.Println(string(infoData))

        // Output:
        // {"Name":"John","Age":18}
}
```

MustEncode

- MustEncodeEncodepanic
-

```
func MustEncode(value interface{}) []byte
```

-

```
func ExampleMustEncode() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        infoData := gjson.MustEncode(info)
        fmt.Println(string(infoData))

        // Output:
        // {"Name":"John","Age":18}
}
```

EncodeString

- EncodeStringvalueJSONstring
-

```
func EncodeString(value interface{}) (string, error)
```

-

```
func ExampleEncodeString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        infoData, _ := gjson.EncodeString(info)
        fmt.Println(infoData)

        // Output:
        // {"Name":"John","Age":18}
}
```

MustEncodeString

- MustEncodeStringvalueJSONstringpanic
-

```
func MustEncodeString(value interface{}) string
```

-

```
func ExampleMustEncodeString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        infoData := gjson.MustEncodeString(info)
        fmt.Println(infoData)

        // Output:
        // {"Name":"John","Age":18}
}
```

Decode

- DecodeJSONdatainterface{} data[]bytestring
-

```
func Decode(data interface{}, options ...Options) (interface{},
error)
```

-

```
func ExampleDecode() {
        jsonContent := `{"name":"john","score":100}`
        info, _ := gjson.Decode([]byte(jsonContent))
        fmt.Println(info)

        // Output:
        // map[name:john score:100]
}
```

DecodeTo

- DecodeToJSONdatainterfacevdata[]bytestringv
-

```
func DecodeTo(data interface{}, v interface{}, options ...Options)
(err error)
```

-

```
func ExampleDecodeTo() {
        type BaseInfo struct {
                Name  string
                Score int
        }

        var info BaseInfo

        jsonContent := "{\"name\":\"john\",\"score\":100}"
        gjson.DecodeTo([]byte(jsonContent), &info)
        fmt.Printf("%+v", info)

        // Output:
        // {Name:john Score:100}
}
```

DecodeToJson

- DecodeToJsonJSONdatajsondata[]bytestring
-

```
func DecodeToJson(data interface{}, options ...Options) (*Json,
error)
```

-

```
func ExampleDecodeToJson() {
        jsonContent := `{"name":"john","score":100}"`
        j, _ := gjson.DecodeToJson([]byte(jsonContent))
        fmt.Println(j.Map())

        // May Output:
        // map[name:john score:100]
}
```

SetSplitChar

- SetSplitChar
-

```
func (j *Json) SetSplitChar(char byte)
```

-

```
func ExampleJson_SetSplitChar() {
        data :=
                `{
        "users" : {
            "count" : 2,
            "list"  : [
                {"name" : "Ming",  "score" : 60},
                {"name" : "John", "score" : 99.5}
            ]
        }
    }`
        if j, err := gjson.DecodeToJson(data); err != nil {
                panic(err)
        } else {
                j.SetSplitChar('#')
                fmt.Println("John Score:", j.Get
("users#list#1#score").Float32())
        }
        // Output:
        // John Score: 99.5
}
```

SetViolenceCheck

- SetViolenceCheck/
-

```
func (j *Json) SetViolenceCheck(enabled bool)
```

-

```
func ExampleJson_SetViolenceCheck() {
        data :=
                `{
        "users" : {
            "count" : 100
        },
        "users.count" : 101
    }`
        if j, err := gjson.DecodeToJson(data); err != nil {
                fmt.Println(err)
        } else {
                j.SetViolenceCheck(true)
                fmt.Println("Users Count:", j.Get("users.count"))
        }
        // Output:
        // Users Count: 101
}
```

ToJson

- ToJson[]byteJSON
-

```
func (j *Json) ToJson() ([]byte, error)
```

-

```
func ExampleJson_ToJson() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonBytes, _ := j.ToJson()
        fmt.Println(string(jsonBytes))

        // Output:
        // {"Age":18,"Name":"John"}
}
```

ToJsonString

- ToJsonStringstringJSON
-

```
func (j *Json) ToJsonString() (string, error)
```

-

```
func ExampleJson_ToJsonString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonStr, _ := j.ToJsonString()
        fmt.Println(jsonStr)

        // Output:
        // {"Age":18,"Name":"John"}
}
```

ToJsonIndent

- ToJsonIndent[]byteJSON
-

```
func (j *Json) ToJsonIndent() ([]byte, error)
```

-

```
func ExampleJson_ToJsonIndent() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonBytes, _ := j.ToJsonIndent()
        fmt.Println(string(jsonBytes))

        // Output:
        //{
        //        "Age": 18,
        //        "Name": "John"
        //}
}
```

ToJsonIndentString

- ToJsonIndentStringstringJSON
-

```
func (j *Json) ToJsonIndentString() (string, error)
```

-

```
func ExampleJson_ToJsonIndentString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonStr, _ := j.ToJsonIndentString()
        fmt.Println(jsonStr)

        // Output:
        //{
        //        "Age": 18,
        //        "Name": "John"
        //}
}
```

MustToJson

- MustToJson[]byteJSONpanic
-

```
func (j *Json) MustToJson() []byte
```

-

```
func ExampleJson_MustToJson() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonBytes := j.MustToJson()
        fmt.Println(string(jsonBytes))

        // Output:
        // {"Age":18,"Name":"John"}
}
```

MustToJsonString

- MustToJsonStringstringJSONpanic
-

```
func (j *Json) MustToJsonString() string
```

-

```
func ExampleJson_MustToJsonString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonStr := j.MustToJsonString()
        fmt.Println(jsonStr)

        // Output:
        // {"Age":18,"Name":"John"}
}
```

MustToJsonIndent

- MustToJsonStringIndent[]byteJSONpanic
-

```
func (j *Json) MustToJsonIndent() []byte
```

-

```
func ExampleJson_MustToJsonIndent() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonBytes := j.MustToJsonIndent()
        fmt.Println(string(jsonBytes))

        // Output:
        //{
        //        "Age": 18,
        //        "Name": "John"
        //}
}
```

MustToJsonIndentString

- MustToJsonStringIndentstringJSONpanic
-

```
func (j *Json) MustToJsonIndentString() string
```

-

```
func ExampleJson_MustToJsonIndentString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonStr := j.MustToJsonIndentString()
        fmt.Println(jsonStr)

        // Output:
        //{
        //        "Age": 18,
        //        "Name": "John"
        //}
}
```

ToXml

- ToXml[]byteXML
-

```
func (j *Json) ToXml(rootTag ...string) ([]byte, error)
```

-

```
func ExampleJson_ToXml() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlBytes, _ := j.ToXml()
        fmt.Println(string(xmlBytes))

        // Output:
        // <doc><Age>18</Age><Name>John</Name></doc>
}
```

ToXmlString

- ToXmlStringstringXML
-

```
func (j *Json) ToXmlString(rootTag ...string) (string, error)
```

-

```
func ExampleJson_ToXmlString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlStr, _ := j.ToXmlString()
        fmt.Println(string(xmlStr))

        // Output:
        // <doc><Age>18</Age><Name>John</Name></doc>
}
```

ToXmlIndent

- ToXmlIndent[]byteXML
-

```
func (j *Json) ToXmlIndent(rootTag ...string) ([]byte, error)
```

-

```
func ExampleJson_ToXmlIndent() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlBytes, _ := j.ToXmlIndent()
        fmt.Println(string(xmlBytes))

        // Output:
        //<doc>
        //        <Age>18</Age>
        //        <Name>John</Name>
        //</doc>
}
```

ToXmlIndentString

- ToXmlIndentStringstringXML
-

```
func (j *Json) ToXmlIndentString(rootTag ...string) (string, error)
```

-

```
func ExampleJson_ToXmlIndentString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlStr, _ := j.ToXmlIndentString()
        fmt.Println(string(xmlStr))

        // Output:
        //<doc>
        //        <Age>18</Age>
        //        <Name>John</Name>
        //</doc>
}
```

MustToXml

- MustToXml[]byteXMLpanic
-

```
func (j *Json) MustToXml(rootTag ...string) []byte
```

-

```
func ExampleJson_MustToXml() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlBytes := j.MustToXml()
        fmt.Println(string(xmlBytes))

        // Output:
        // <doc><Age>18</Age><Name>John</Name></doc>
}
```

MustToXmlString

- MustToXmlStringstringXMLpanic
-

```
func (j *Json) MustToXmlString(rootTag ...string) string
```

-

```
func ExampleJson_MustToXmlString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlStr := j.MustToXmlString()
        fmt.Println(string(xmlStr))

        // Output:
        // <doc><Age>18</Age><Name>John</Name></doc>
}
```

MustToXmlIndent

- MustToXmlStringIndent[]byteXMLpanic
-

```
func (j *Json) MustToXmlIndent(rootTag ...string) []byte
```

-

```
func ExampleJson_MustToXmlIndent() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlBytes := j.MustToXmlIndent()
        fmt.Println(string(xmlBytes))

        // Output:
        //<doc>
        //        <Age>18</Age>
        //        <Name>John</Name>
        //</doc>
}
```

MustToXmlIndentString

- MustToXmlStringIndentStringstringXMLpanic
-

```
func (j *Json) MustToXmlIndentString(rootTag ...string) string
```

-

```
func ExampleJson_MustToXmlIndentString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        xmlStr := j.MustToXmlIndentString()
        fmt.Println(string(xmlStr))

        // Output:
        //<doc>
        //        <Age>18</Age>
        //        <Name>John</Name>
        //</doc>
}
```

ToYaml

- ToYaml[]byteYAML
-

```
func (j *Json) ToYaml() ([]byte, error)
```

-

```
func ExampleJson_ToYaml() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        YamlBytes, _ := j.ToYaml()
        fmt.Println(string(YamlBytes))

        // Output:
        //Age: 18
        //Name: John
}
```

ToYamlIndent

- ToYamlIndent[]byteYAML
-

```
func (j *Json) ToYamlIndent(indent string) ([]byte, error)
```

-

```
func ExampleJson_ToYamlIndent() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        YamlBytes, _ := j.ToYamlIndent("")
        fmt.Println(string(YamlBytes))

        // Output:
        //Age: 18
        //Name: John
}
```

ToYamlString

- ToYamlStringstringYAML
-

```
func (j *Json) ToYamlString() (string, error)
```

-

```
func ExampleJson_ToYamlString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        YamlStr, _ := j.ToYamlString()
        fmt.Println(string(YamlStr))

        // Output:
        //Age: 18
        //Name: John
}
```

MustToYaml

- MustToYaml[]byteYAMLpanic
-

```
func (j *Json) MustToYaml() []byte
```

-

```
func ExampleJson_MustToYaml() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        YamlBytes := j.MustToYaml()
        fmt.Println(string(YamlBytes))

        // Output:
        //Age: 18
        //Name: John
}
```

MustToYamlString

- MustToYamlStringstringYAMLpanic
-

```
func (j *Json) MustToYamlString() string
```

-

```
func ExampleJson_MustToYamlString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        YamlStr := j.MustToYamlString()
        fmt.Println(string(YamlStr))

        // Output:
        //Age: 18
        //Name: John
}
```

ToToml

- ToToml[]byteTOML
-

```
func (j *Json) ToToml() ([]byte, error)
```

-

```
func ExampleJson_ToToml() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        TomlBytes, _ := j.ToToml()
        fmt.Println(string(TomlBytes))

        // Output:
        //Age = 18
        //Name = "John"
}
```

ToTomlString

- ToTomlStringstringTOML
-

```
func (j *Json) ToTomlString() (string, error)
```

-

```
func ExampleJson_ToTomlString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        TomlStr, _ := j.ToTomlString()
        fmt.Println(string(TomlStr))

        // Output:
        //Age = 18
        //Name = "John"
}
```

MustToToml

- MustToToml[]byteTOMLpanic
-

```
func (j *Json) MustToToml() []byte
```

-

```
func ExampleJson_MustToToml() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        TomlBytes := j.MustToToml()
        fmt.Println(string(TomlBytes))

        // Output:
        //Age = 18
        //Name = "John"
}
```

MustToTomlString

- MustToTomlStringstringTOMLpanic
-

```
func (j *Json) MustToTomlString() string
```

-

```
func ExampleJson_MustToTomlString() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        TomlStr := j.MustToTomlString()
        fmt.Println(string(TomlStr))

        // Output:
        //Age = 18
        //Name = "John"
}
```

ToIni

- ToIni[]byteINI
-

```
func (j *Json) ToIni() ([]byte, error)
```

-

```
func ExampleJson_ToIni() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        IniBytes, _ := j.ToIni()
        fmt.Println(string(IniBytes))

        // May Output:
        //Name=John
        //Age=18
}
```

ToIniString

- ToIniStringstringINI
-

```
func (j *Json) ToIniString() (string, error)
```

-

```
func ExampleJson_ToIniString() {
        type BaseInfo struct {
                Name string
        }

        info := BaseInfo{
                Name: "John",
        }

        j := gjson.New(info)
        IniStr, _ := j.ToIniString()
        fmt.Println(string(IniStr))

        // Output:
        //Name=John
}
```

MustToIni

- MustToIni[]byteINIpanic
-

```
func (j *Json) MustToIni() []byte
```

-

```
func ExampleJson_MustToIni() {
        type BaseInfo struct {
                Name string
        }

        info := BaseInfo{
                Name: "John",
        }

        j := gjson.New(info)
        IniBytes := j.MustToIni()
        fmt.Println(string(IniBytes))

        // Output:
        //Name=John
}
```

MustToIniString

- MustToIniStringstringINIpanic
-

```
func (j *Json) MustToIniString() string
```

-

```
func ExampleJson_MustToIniString() {
        type BaseInfo struct {
                Name string
        }

        info := BaseInfo{
                Name: "John",
        }

        j := gjson.New(info)
        IniStr := j.MustToIniString()
        fmt.Println(string(IniStr))

        // Output:
        //Name=John
}
```

MarshalJSON

- MarshalJSONjson.MarshalMarshalJSON
-

```
func (j Json) MarshalJSON() ([]byte, error)
```

-

```
func ExampleJson_MarshalJSON() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        jsonBytes, _ := j.MarshalJSON()
        fmt.Println(string(jsonBytes))

        // Output:
        // {"Age":18,"Name":"John"}
}
```

UnmarshalJSON

- UnmarshalJSON json.Unmarshal UnmarshalJSON
-

```
func (j *Json) UnmarshalJSON(b []byte) error
```

-

```
func ExampleJson_UnmarshalJSON() {
        jsonStr := `{"Age":18,"Name":"John"}`

        j := gjson.New("")
        j.UnmarshalJSON([]byte(jsonStr))
        fmt.Println(j.Map())

        // Output:
        // map[Age:18 Name:John]
}
```

UnmarshalValue

- UnmarshalValueJson
-

```
func (j *Json) UnmarshalValue(value interface{}) error
```

-

```
func ExampleJson_UnmarshalValue_Yaml() {
        yamlContent :=
                `base:
  name: john
  score: 100`

        j := gjson.New("")
        j.UnmarshalValue([]byte(yamlContent))
        fmt.Println(j.Var().String())

        // Output:
        // {"base":{"name":"john","score":100}}
}
```

```
func ExampleJson_UnmarshalValue_Xml() {
        xmlStr := `<?xml version="1.0" encoding="UTF-8"?
><doc><name>john</name><score>100</score></doc>`

        j := gjson.New("")
        j.UnmarshalValue([]byte(xmlStr))
        fmt.Println(j.Var().String())

        // Output:
        // {"doc":{"name":"john","score":"100"}}
}
```

MapStrAny

- MapStrAnyMapStrAny()
-

```
func (j *Json) MapStrAny() map[string]interface{}
```

-

```
func ExampleJson_MapStrAny() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        fmt.Println(j.MapStrAny())

        // Output:
        // map[Age:18 Name:John]
}
```

Interfaces

- InterfacesInterfaces()
-

```
func (j *Json) Interfaces() []interface{}
```

-

```
func ExampleJson_Interfaces() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        infoList := []BaseInfo{
                BaseInfo{
                        Name: "John",
                        Age:  18,
                },
                BaseInfo{
                        Name: "Tom",
                        Age:  20,
                },
        }

        j := gjson.New(infoList)
        fmt.Println(j.Interfaces())

        // Output:
        // [{John 18} {Tom 20}]
}
```

Interface

- InterfaceJson
-

```
func (j *Json) Interface() interface{}
```

-

```
func ExampleJson_Interface() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        fmt.Println(j.Interface())

        var nilJ *gjson.Json = nil
        fmt.Println(nilJ.Interface())

        // Output:
        // map[Age:18 Name:John]
        // <nil>
}
```

Var

- `Var*gvar.VarJson`
-

```
func (j *Json) Var() *gvar.Var
```

-

```
func ExampleJson_Var() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        fmt.Println(j.Var().String())
        fmt.Println(j.Var().Map())

        // Output:
        // {"Age":18,"Name":"John"}
        // map[Age:18 Name:John]
}
```

IsNil

- `IsNilJsonnil`
-

```
func (j *Json) IsNil() bool
```

-

```
func ExampleJson_IsNil() {
        data1 := []byte(`{"n":123456789, "m":{"k":"v"}, "a":[1,2,3]}
`)
        data2 := []byte(`{"n":123456789, "m":{"k":"v"}, "a":[1,2,3]`)

        j1, _ := gjson.LoadContent(data1)
        fmt.Println(j1.IsNil())

        j2, _ := gjson.LoadContent(data2)
        fmt.Println(j2.IsNil())

        // Output:
        // false
        // true
}
```

Get

- Getpatternpattern"."Jsonpatternnil
-

```
func (j *Json) Get(pattern string, def ...interface{}) *gvar.Var
```

-

```
func ExampleJson_Get() {
        data :=
                `{
        "users" : {
            "count" : 1,
            "array" : ["John", "Ming"]
        }
    }`

        j, _ := gjson.LoadContent(data)
        fmt.Println(j.Get("."))
        fmt.Println(j.Get("users"))
        fmt.Println(j.Get("users.count"))
        fmt.Println(j.Get("users.array"))

        var nilJ *gjson.Json = nil
        fmt.Println(nilJ.Get("."))

        // Output:
        // {"users":{"array":["John","Ming"],"count":1}}
        // {"array":["John","Ming"],"count":1}
        // 1
        // ["John","Ming"]
}
```

GetJson

- GetJsonpatternJson
-

```
func (j *Json) GetJson(pattern string, def ...interface{}) *Json
```

-

```
func ExampleJson_GetJson() {
        data :=
                `{
        "users" : {
            "count" : 1,
            "array" : ["John", "Ming"]
        }
    }`

        j, _ := gjson.LoadContent(data)

        fmt.Println(j.GetJson("users.array").Array())

        // Output:
        // [John Ming]
}
```

GetJsons

- GetJsonspatternJson
-

```
func (j *Json) GetJsons(pattern string, def ...interface{}) []*Json
```

-

```
func ExampleJson_GetJsons() {
        data :=
                `{
        "users" : {
            "count" : 3,
            "array" : [{"Age":18,"Name":"John"}, {"Age":20,"Name":"
Tom"}]
        }
    }`

        j, _ := gjson.LoadContent(data)

        jsons := j.GetJsons("users.array")
        for _, json := range jsons {
                fmt.Println(json.Interface())
        }

        // Output:
        // map[Age:18 Name:John]
        // map[Age:20 Name:Tom]
}
```

GetJsonMap

- GetJsonMappatternJsonmap
-

```
func (j *Json) GetJsonMap(pattern string, def ...interface{}) map
[string]*Json
```

-

```
func ExampleJson_GetJsonMap() {
        data :=
                `{
        "users" : {
            "count" : 1,
                        "array" : {
                                "info" : {"Age":18,"Name":"John"},
                                "addr" : {"City":"Chengdu","
Company":"Tencent"}
                        }
        }
    }`

        j, _ := gjson.LoadContent(data)

        jsonMap := j.GetJsonMap("users.array")

        for _, json := range jsonMap {
                fmt.Println(json.Interface())
        }

        // May Output:
        // map[City:Chengdu Company:Tencent]
        // map[Age:18 Name:John]
}
```

Set

- Setpattern '.'
-

```
func (j *Json) Set(pattern string, value interface{}) error
```

-

```
func ExampleJson_Set() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        j.Set("Addr", "ChengDu")
        j.Set("Friends.0", "Tom")
        fmt.Println(j.Var().String())

        // Output:
        // {"Addr":"ChengDu","Age":18,"Friends":["Tom"],"Name":"
John"}
}
```

MustSet

- MustSetSetpanic
-

```
func (j *Json) MustSet(pattern string, value interface{})
```

-

```
func ExampleJson_MustSet() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        j.MustSet("Addr", "ChengDu")
        fmt.Println(j.Var().String())

        // Output:
        // {"Addr":"ChengDu","Age":18,"Name":"John"}
}
```

Remove

- Removepattern '.'
-

```
func (j *Json) Remove(pattern string) error
```

-

```
func ExampleJson_Remove() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        j.Remove("Age")
        fmt.Println(j.Var().String())

        // Output:
        // {"Name":"John"}
}
```

MustRemove

- MustRemoveRemovepanic
-

```
func (j *Json) MustRemove(pattern string)
```

-

```
func ExampleJson_MustRemove() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        j.MustRemove("Age")
        fmt.Println(j.Var().String())

        // Output:
        // {"Name":"John"}
}
```

Contains

- Containspattern
-

```
func (j *Json) Contains(pattern string) bool
```

-

```
func ExampleJson_Contains() {
        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{
                Name: "John",
                Age:  18,
        }

        j := gjson.New(info)
        fmt.Println(j.Contains("Age"))
        fmt.Println(j.Contains("Addr"))

        // Output:
        // true
        // false
}
```

Len

- Lenpattern / patternslicemap -1
-

```
func (j *Json) Len(pattern string) int
```

-

```
func ExampleJson_Len() {
        data :=
                `{
        "users" : {
            "count" : 1,
                        "nameArray" : ["Join", "Tom"],
                        "infoMap" : {
                                "name" : "Join",
                                "age" : 18,
                                "addr" : "ChengDu"
                        }
        }
    }`

        j, _ := gjson.LoadContent(data)

        fmt.Println(j.Len("users.nameArray"))
        fmt.Println(j.Len("users.infoMap"))

        // Output:
        // 2
        // 3
}
```

Append

- AppendpatternJson patternslice
-

```
func (j *Json) Append(pattern string, value interface{}) error
```

-

```
func ExampleJson_Append() {
        data :=
                `{
        "users" : {
            "count" : 1,
            "array" : ["John", "Ming"]
        }
    }`

        j, _ := gjson.LoadContent(data)

        j.Append("users.array", "Lily")

        fmt.Println(j.Get("users.array").Array())

        // Output:
        // [John Ming Lily]
}
```

MustAppend

- MustAppendAppendpanic
-

```
func (j *Json) MustAppend(pattern string, value interface{})
```

-

```
func ExampleJson_MustAppend() {
        data :=
                `{
        "users" : {
            "count" : 1,
            "array" : ["John", "Ming"]
        }
    }`

        j, _ := gjson.LoadContent(data)

        j.MustAppend("users.array", "Lily")

        fmt.Println(j.Get("users.array").Array())

        // Output:
        // [John Ming Lily]
}
```

Map

- `MapJsonmap[string]interface{}` `nil`
-

```
func (j *Json) Map() map[string]interface{}
```

-

```
func ExampleJson_Map() {
        data :=
                `{
        "users" : {
            "count" : 1,
            "info" : {
                                "name" : "John",
                                "age" : 18,
                                "addr" : "ChengDu"
                        }
        }
    }`

        j, _ := gjson.LoadContent(data)

        fmt.Println(j.Get("users.info").Map())

        // Output:
        // map[addr:ChengDu age:18 name:John]
}
```

Array

- `ArrayJson[]interface{}` `nil`
-

```
func (j *Json) Array() []interface{}
```

-

```
func ExampleJson_Array() {
        data :=
                `{
        "users" : {
            "count" : 1,
            "array" : ["John", "Ming"]
        }
    }`

        j, _ := gjson.LoadContent(data)

        fmt.Println(j.Get("users.array"))

        // Output:
        // ["John","Ming"]
}
```

Scan

- ScanStructStructspointer
-

```
func (j *Json) Scan(pointer interface{}, mapping ...map[string]
string) error
```

-

```
func ExampleJson_Scan() {
        data := `{"name":"john","age":"18"}`

        type BaseInfo struct {
                Name string
                Age  int
        }

        info := BaseInfo{}

        j, _ := gjson.LoadContent(data)
        j.Scan(&info)

        fmt.Println(info)

        // May Output:
        // {john 18}
}
```

Dump

- DumpJson
-

```
func (j *Json) Dump()
```

-

```
func ExampleJson_Dump() {
        data := `{"name":"john","age":"18"}`

        j, _ := gjson.LoadContent(data)
        j.Dump()

        // May Output:
        //{
        //        "name": "john",
        //        "age":  "18",
        //}
}
```