
LightJSON Documentation

Release 0.1.0

Shenggan

Jan 26, 2018

Contents

1	Overview	1
2	Build and Install	3
3	Usage	5
3.1	Class Style API	5
3.2	C Style API	5
4	APIs	7
4.1	lightjson.h	7

CHAPTER 1

Overview

LightJSON is a lightweight and modern Json C++ library

The function:

- parse json string
- access the edit json content
- generate json string

The highlight feature:

- small but complete
- using many new feature in C++11
- use googletest for unit test
- LightJSON provide two style APIs
 - Class Style API
 - C Style API

CHAPTER 2

Build and Install

The build of the LightJSON need cmake.

```
bash ./scripts/build.sh
```

You can run the UnitTest to validate the success of the build.

```
bash ./scripts/test.sh
```


3.1 Class Style API

```
#include <iostream>
#include <string>
#include "lightjson.h"

int main() {
    ljson::ljson_value v;
    ljson_init(&v);

    std::string str("{ \"1\" : 1 }");
    ljson_parse(&v, str);

    std::string str2;
    ljson_stringify(&v, str2);
    std::cout << str2 << std::endl;

    ljson_free(&v);
    return 0;
}
```

3.2 C Style API

```
#include <iostream>
#include <string>
#include "lightjson.h"

int main() {
    ljson::ljson_value v;
    ljson_init(&v);
```

```
std::string str("{ \"1\" : 1 }");
ljson_parse(&v, str);

std::string str2;
ljson_stringify(&v, str2);
std::cout << str2 << std::endl;

ljson_free(&v);
return 0;
}
```

4.1 lightjson.h

The head file of LightJSON.

Author Shenggan

namespace ljson

Typedefs

```
typedef struct ljson_value ljson_value  
typedef struct ljson_member ljson_member
```

Enums

```
enum ljson_type  
    the basic type of the json struct  
  
    Values:  
  
    LJSON_NULL  
  
    LJSON_FALSE  
  
    LJSON_TRUE  
  
    LJSON_NUMBER  
  
    LJSON_STRING  
  
    LJSON_ARRAY  
  
    LJSON_OBJECT
```

enum ljson_state
the type of the results or error

Values:

- LJSON_PARSE_OK** = 0
- LJSON_STRINGIFY_OK**
- LJSON_PARSE_EXPECT_VALUE**
- LJSON_PARSE_INVALID_VALUE**
- LJSON_PARSE_ROOT_NOT_SINGULAR**
- LJSON_PARSE_NUMBER_TOO_BIG**
- LJSON_PARSE_MISS_QUOTATION_MARK**
- LJSON_PARSE_INVALID_STRING_ESCAPE**
- LJSON_PARSE_INVALID_STRING_CHAR**
- LJSON_PARSE_INVALID_UNICODE_HEX**
- LJSON_PARSE_INVALID_UNICODE_SURROGATE**
- LJSON_PARSE_MISS_COMMA_OR_SQUARE_BRACKET**
- LJSON_PARSE_MISS_KEY**
- LJSON_PARSE_MISS_COLON**
- LJSON_PARSE_MISS_COMMA_OR_CURLY_BRACKET**

Functions

void **ljson_free** (*ljson_value* *v)
free the memory of a *ljson_value*, use it if you will not use it or initailize it again

Parameters

- v: the pointer of *ljson_value* you want to free

void **ljson_init** (*ljson_value* *v)
initailize a *ljson_value*, use it after declaration

Parameters

- v: the pointer of *ljson_value* you want to initailize

void **setNull** (*ljson_value* *v)
the same as *ljson_free(ljson_value* v)*

void **setNull** (*ljson_value* &v)

int **ljson_parse** (*ljson_value* *v, **const** char *json)
parse a string to get the *ljson_value*

Return ljson_state

Parameters

- *v*: the pointer of *ljson_value* you want to store the result of parse
- *json*: the string you want to parse

int **ljson_parse** (*ljson_value* **v*, const std::string &*json*)
 parse a string to get the *ljson_value*

Return *ljson_state*

Parameters

- *v*: the pointer of *ljson_value* you want to store the result of parse
- *json*: the string you want to parse

int **ljson_stringify** (const *ljson_value* **v*, std::string &*json*)
ljson_value *v* to get the string os the json

Return *ljson_state*

Parameters

- *v*: the pointer of *ljson_value* you want to stringify
- *json*: the string you want to store the result

int **ljson_stringify** (const *ljson_value* **v*, char **json*)
ljson_value *v* to get the string os the json

Return *ljson_state*

Parameters

- *v*: the pointer of *ljson_value* you want to stringify
- *json*: the string you want to store the result

ljson_type **getType** (const *ljson_value* **v*)

ljson_type **getType** (const *ljson_value* &*v*)

void **setNumber** (*ljson_value* **v*, double *n*)

void **setNumber** (*ljson_value* &*v*, double *n*)

double **getNumber** (const *ljson_value* **v*)

double **getNumber** (const *ljson_value* &*v*)

void **setBool** (*ljson_value* **v*, *ljson_type* *b*)

void **setBool** (*ljson_value* **v*, bool *b*)

bool **getBool** (const *ljson_value* **v*)

void **setBool** (*ljson_value* &*v*, *ljson_type* *b*)

void **setBool** (*ljson_value* &*v*, bool *b*)

bool **getBool** (const *ljson_value* &*v*)

```
void setString (ljson_value *v, const char *s, size_t len)

void setString (ljson_value *v, const std::string &s)

std::string &getString (const ljson_value *v)

size_t getStringLength (const ljson_value *v)

void setString (ljson_value &v, const char *s, size_t len)

void setString (ljson_value &v, const std::string &s)

std::string &getString (const ljson_value &v)

size_t getStringLength (const ljson_value &v)

void setArray (ljson_value *v, const std::vector<ljson_value> &vec, bool deep_copy = 1)

std::vector<ljson_value> &getArray (const ljson_value *v)

ljson_value &getArrayElement (const ljson_value *v, const size_t index)

void setArrayElement (ljson_value *v, const size_t index, const ljson_value &content)

size_t getArraySize (const ljson_value *v)

void setArray (ljson_value &v, const std::vector<ljson_value> &vec, bool deep_copy = 1)

std::vector<ljson_value> &getArray (const ljson_value &v)

ljson_value &getArrayElement (const ljson_value &v, const size_t index)

void setArrayElement (ljson_value &v, const size_t index, const ljson_value &content)

size_t getArraySize (const ljson_value &v)

void setObject (ljson_value *v, const std::map<std::string, ljson_value> &vec, bool deep_copy = 1)

bool objectFindKey (const ljson_value *v, const std::string &mkey)

bool objectFindKey (const ljson_value &v, const std::string &mkey)

std::map<std::string, ljson_value> &getObject (const ljson_value *v)

ljson_value &getObjElement (const ljson_value *v, const std::string &key)

void setObjElement (ljson_value *v, const std::string key, const ljson_value &content)

size_t getObjectSize (const ljson_value *v)

ljson_value &objectAccess (ljson_value *v, const std::string &mkey)

void setObject (ljson_value &v, const std::map<std::string, ljson_value> &vec, bool deep_copy =
    1)

ljson_value &getObjElement (const ljson_value &v, const std::string key)

void setObjElement (ljson_value &v, const std::string key, const ljson_value &content)

size_t getObjectSize (const ljson_value &v)

ljson_value &objectAccess (ljson_value &v, const std::string &mkey)
```

```

void ljson_reset (ljson_value *v_old, const ljson_value &v_new)

void ljson_reset (ljson_value *v_old, const ljson_value *v_new)

static int ljson_parse_value (ljson_context *c, ljson_value *v)

void expect_char (ljson_context *c, char ch)

bool isdigit1to9 (char ch)

static void ljson_parse_whitespace (ljson_context *c)

static int ljson_parse_literal (ljson_context *c, ljson_value *v, const char *literal, ljson_type type)

static int ljson_parse_number (ljson_context *c, ljson_value *v)

static const char *ljson_parse_hex4 (const char *p, unsigned *u)

static void ljson_encode_utf8 (std::string &s, unsigned u)

static int ljson_parse_string_raw (ljson_context *c, std::string &cache_string)

static int ljson_parse_string (ljson_context *c, ljson_value *v)

static int ljson_parse_array (ljson_context *c, ljson_value *v)

static int ljson_parse_object (ljson_context *c, ljson_value *v)

static void ljson_stringify_string (std::string &str, const std::string &json_str)

static int ljson_stringify_value (const ljson_value *v, std::string &str)

std::map<std::string, ljson_value> &getObject (const ljson_value &v)

class Document
    #include <lightjson.h> Inherits from ljson::Value

```

Public Functions

```

Document ()

~Document ()

int Parse (std::string &json)

struct ljson_context
    #include <lightjson.h>

```

Public Members

```

const char *json

struct ljson_member
    #include <lightjson.h> the struct of the member of json object

```

Public Members

std::string **key**

ljson_value **value**

struct ljson_object

#include <lightjson.h> the struct of the json object

Public Members

char ***name**

ljson_value **value**

struct ljson_value

#include <lightjson.h> the inner struct of a json, can present all kind of ljson_type

Public Functions

void **free** ()

void **copyfrom** (const *ljson_value* ©)

Public Members

union ljson::ljson_value::__data data

data part of *ljson_value*

ljson_type **type**

typr of this *ljson_value*

union __data

#include <lightjson.h>

Public Members

std::map<std::string, *ljson_value*> ***mobject**
object

std::string ***mstring**
string

std::vector<*ljson_value*> ***marray**
array

double **mdouble**
number

class Value

#include <lightjson.h> Subclassed by *ljson::Document*

Public Functions

Value ()

Value (*ljson_value* *v)

~Value ()

Value **operator []** (const std::string key) const

Value **operator []** (const size_t index) const

ljson_value ***GetValue** () const

void **SetValue** (const *Value* &content)

void **SetNumber** (const double a_num)

double **GetNumber** () const

void **SetBool** (const bool a_bool)

bool **GetBool** () const

void **SetString** (const std::string &a_str)

std::string &**GetString** () const

void **SetArrayElement** (const size_t index, const *Value* &content)

Value **GetArrayElement** (const size_t index) const

void **SetObjectElement** (const std::string &key, const *Value* &content)

Value **GetObjElement** (const std::string &key) const

std::string **cpp_str** () const

Protected Attributes

ljson_value *mvalue

Friends

std::ostream &**operator<<** (std::ostream &out, const Value &v)

L

- ljson (C++ type), 7
- ljson::Document (C++ class), 11
- ljson::Document::~~Document (C++ function), 11
- ljson::Document::Document (C++ function), 11
- ljson::Document::Parse (C++ function), 11
- ljson::expect_char (C++ function), 11
- ljson::getArray (C++ function), 10
- ljson::getArrayElement (C++ function), 10
- ljson::getArraySize (C++ function), 10
- ljson::getBool (C++ function), 9
- ljson::getNumber (C++ function), 9
- ljson::getObject (C++ function), 10, 11
- ljson::getObjectSize (C++ function), 10
- ljson::getObjElement (C++ function), 10
- ljson::getString (C++ function), 10
- ljson::getStringLength (C++ function), 10
- ljson::getType (C++ function), 9
- ljson::isdigit1to9 (C++ function), 11
- ljson::LJSON_ARRAY (C++ enumerator), 7
- ljson::ljson_context (C++ class), 11
- ljson::ljson_context::json (C++ member), 11
- ljson::ljson_encode_utf8 (C++ function), 11
- ljson::LJSON_FALSE (C++ enumerator), 7
- ljson::ljson_free (C++ function), 8
- ljson::ljson_init (C++ function), 8
- ljson::ljson_member (C++ class), 11
- ljson::ljson_member (C++ type), 7
- ljson::ljson_member::key (C++ member), 12
- ljson::ljson_member::value (C++ member), 12
- ljson::LJSON_NULL (C++ enumerator), 7
- ljson::LJSON_NUMBER (C++ enumerator), 7
- ljson::ljson_object (C++ class), 12
- ljson::LJSON_OBJECT (C++ enumerator), 7
- ljson::ljson_object::name (C++ member), 12
- ljson::ljson_object::value (C++ member), 12
- ljson::ljson_parse (C++ function), 8, 9
- ljson::ljson_parse_array (C++ function), 11
- ljson::LJSON_PARSE_EXPECT_VALUE (C++ enumerator), 8
- ljson::ljson_parse_hex4 (C++ function), 11
- ljson::LJSON_PARSE_INVALID_STRING_CHAR (C++ enumerator), 8
- ljson::LJSON_PARSE_INVALID_STRING_ESCAPE (C++ enumerator), 8
- ljson::LJSON_PARSE_INVALID_UNICODE_HEX (C++ enumerator), 8
- ljson::LJSON_PARSE_INVALID_UNICODE_SURROGATE (C++ enumerator), 8
- ljson::LJSON_PARSE_INVALID_VALUE (C++ enumerator), 8
- ljson::ljson_parse_literal (C++ function), 11
- ljson::LJSON_PARSE_MISS_COLON (C++ enumerator), 8
- ljson::LJSON_PARSE_MISS_COMMA_OR_CURLY_BRACKET (C++ enumerator), 8
- ljson::LJSON_PARSE_MISS_COMMA_OR_SQUARE_BRACKET (C++ enumerator), 8
- ljson::LJSON_PARSE_MISS_KEY (C++ enumerator), 8
- ljson::LJSON_PARSE_MISS_QUOTATION_MARK (C++ enumerator), 8
- ljson::ljson_parse_number (C++ function), 11
- ljson::LJSON_PARSE_NUMBER_TOO_BIG (C++ enumerator), 8
- ljson::ljson_parse_object (C++ function), 11
- ljson::LJSON_PARSE_OK (C++ enumerator), 8
- ljson::LJSON_PARSE_ROOT_NOT_SINGULAR (C++ enumerator), 8
- ljson::ljson_parse_string (C++ function), 11
- ljson::ljson_parse_string_raw (C++ function), 11
- ljson::ljson_parse_value (C++ function), 11
- ljson::ljson_parse_whitespace (C++ function), 11
- ljson::ljson_reset (C++ function), 10, 11
- ljson::ljson_state (C++ type), 7
- ljson::LJSON_STRING (C++ enumerator), 7
- ljson::ljson_stringify (C++ function), 9
- ljson::LJSON_STRINGIFY_OK (C++ enumerator), 8
- ljson::ljson_stringify_string (C++ function), 11
- ljson::ljson_stringify_value (C++ function), 11

`ljson::LJSON_TRUE` (C++ enumerator), 7
`ljson::ljson_type` (C++ type), 7
`ljson::ljson_value` (C++ class), 12
`ljson::ljson_value` (C++ type), 7
`ljson::ljson_value::__data` (C++ type), 12
`ljson::ljson_value::__data::marray` (C++ member), 12
`ljson::ljson_value::__data::mdouble` (C++ member), 12
`ljson::ljson_value::__data::mobject` (C++ member), 12
`ljson::ljson_value::__data::mstring` (C++ member), 12
`ljson::ljson_value::copyfrom` (C++ function), 12
`ljson::ljson_value::data` (C++ member), 12
`ljson::ljson_value::free` (C++ function), 12
`ljson::ljson_value::type` (C++ member), 12
`ljson::objectAccess` (C++ function), 10
`ljson::objectFindKey` (C++ function), 10
`ljson::setArray` (C++ function), 10
`ljson::setArrayElement` (C++ function), 10
`ljson::setBool` (C++ function), 9
`ljson::setNull` (C++ function), 8
`ljson::setNumber` (C++ function), 9
`ljson::setObject` (C++ function), 10
`ljson::setObjElement` (C++ function), 10
`ljson::setString` (C++ function), 9, 10
`ljson::Value` (C++ class), 12
`ljson::Value::~~Value` (C++ function), 13
`ljson::Value::cpp_str` (C++ function), 13
`ljson::Value::GetArrayElement` (C++ function), 13
`ljson::Value::GetBool` (C++ function), 13
`ljson::Value::GetNumber` (C++ function), 13
`ljson::Value::GetObjElement` (C++ function), 13
`ljson::Value::GetString` (C++ function), 13
`ljson::Value::GetValue` (C++ function), 13
`ljson::Value::mvalue` (C++ member), 13
`ljson::Value::operator[]` (C++ function), 13
`ljson::Value::SetArrayElement` (C++ function), 13
`ljson::Value::SetBool` (C++ function), 13
`ljson::Value::SetNumber` (C++ function), 13
`ljson::Value::SetObjectElement` (C++ function), 13
`ljson::Value::SetString` (C++ function), 13
`ljson::Value::SetValue` (C++ function), 13
`ljson::Value::Value` (C++ function), 13

O

`operator<<` (C++ function), 13