

**claujson**

vztpv@naver.com

# ClauJson

- experimental parallel json parser (using simd and multi-thread)
- -scj3 (using modified simdjson and clau-parser code)
- use **mimalloc**, ... for performance.

# init

first, call `init` function.

# Value

class

json data : 3, 4.5, "hello", [ 1, 2, 3 ], and { "key": "value" }, ...

Value`s destructor **do not delete object or array**.  
(if Value is object or array)

if you want to delete object or array, call `claujson::clean` function.

use `std::move`! no copy constructor, no assignment.

if need copy, then use `clone()` method.

# Array

class

```
std::vector<Value> data_vec;
```

Array's Destructor **delete** child! (if child is array or object)

# Object

class

```
std::vector<Value> key_vec; // must string!
```

```
std::vector<Value> val_vec;
```

Object's Destructor **delete** child! (if val is array or object)

clean

function

for Value.

# check dup of key

- in object.



## parse, parse\_str

- `thr_num <- 0` : use all thread of cpu.
- not thread-safe
- `parse(_from_file)`

## save, save\_parallel

- save - slow
- save\_parallel - not thread-safe; when use this function, must not access data.

# log

- `claujson::log.no_print();`

# check

- A. no check dup of key when parsing.
- B. Value j("test string"); // utf-8 check, unicode check. etc..
- C. object <- because of A, like array, object also access using index. (order by input)