JsonC Manual

Changelog	
Description	**Date**
Initial version	23-May-2020

- ## Datatypes

1. **json_dtype_t**

   An enumeration definition that represents the json data types.
   Following are the data types in jsonc:

   - json_dtype_ltr: true/false/null
   - json_dtype_num: numbers. e.g. 123, 0.123, -0.1, 23e+3
   - json_dtype_str: strings. e.g. “any valid string”
   - json_dtype_arr: json array. [...]
   - json_dtype_obj: json object. {...}

2. **json_t**

   This represents a json value. A json value can be any object of any json datatype.

3. **json_arr_t**

   This represents a json array.

4. **json_obj_t**

   This represents a json obj

5. **NOTE:**

   json_dtype_ltr, json_dtype_num, json_dtype_str are implemented using C string.

- ## Functions on json_t

1. **json_t json_value_create(void *value, json_dtype_t dtype)**

   This function will create a json_t object.

   Parameters:
   value: A pointer to an object of any json_dtype_t.
   dtype: The datatype of the object.

   Returns: a json_t object on success, NULL if failed.

2. **void json_value_free(json_t value)**

   Free the json_t object created by json_value_create.

   Parameter:
   value: A json_t object previously created by json_value_create.

   Returns: void

3. **json_obj_t json_value_to_obj(json_t val)**

   Get the json object in json value.

   Parameters:
   val: A json value.

   Returns: json_obj_t object on success, NULL if failed.
   If the operation fails because of datatype missmatch i.e. if the datatype of val is not json_dtype_obj, json_error is set to json_error_dtype_mismatch_access.

4. **json_arr_t json_value_to_arr(json_t val)**

   Get the json array in json value.
   Parameters:
   val: A json_t object.
   Returns: json_arr_t object on success, NULL if failed.

   If the operation fails because of datatype missmatch i.e. if the datatype of val is not json_dtype_arr, json_error is set to json_error_dtype_mismatch_access.

5. **char *json_value_to_str(json_t val)**

   Get the C string in json value.

   Parameters:
   val: A json_t object.

   Returns: A pointer to the data in val on success, NULL if failed.

   If the operation fails because of datatype missmatch i.e. if the datatype of val is json_dtype_obj or json_dtype_arr, json_error is set to json_error_dtype_mismatch_access.

6. **void *json_value_to_val(json_t val)**

   Get the pointer to the data in val.

   Parameters:
   val: A json_t object

   Returns: A pointer to the data in val.

   Note: This returned value must be type cast to the proper datatype.

7. **bool json_value_is_true(json_t val)**

   Checks whether a json_t object is true.

   Parameters:
   val: A json_t object

   Returns: true if val is “true”, false otherwise.

8. **bool json_value_is_false(json_t val)**

   Checks whether a json_t object is false.

   Parameters:
   val: A json_t object.

   Returns: true if val is “false”, false otherwise

9. **bool json_value_is_null(json_t val)**

   Checks whether a json_t object is null.

   Parameters:
   val: A json_t object.

   Returns: true if val is “null”, false otherwise

- **Functions on json_arr_t**

1. **void json_arr_init(json_arr_t *json_arr_p)**

   Initialises a json array. It does some initial memory allocation for the array.

   Parameters:
   json_arr_p: A json_arr_t pointer.

   Returns: void

2. **int json_arr_push_back(json_arr_t *json_arr_p, json_t value)**

   Insert a json value to the array.

   Parameters:
   json_arr_p: A pointer to an initialised json_arr_t object.
   value: A json_t object to be inserted to the array.

   Returns: Index of the inserted value, -1 on failed

3. **int json_arr_len(json_arr_t json_arr)**

   Get the length of a json array.

   Parameters:
   json_arr: A json_arr_t object

   Returns: The length of the array

4. **json_t json_arr_get(const json_arr_t json_arr, int idx)**

   Get a json value from the json array.

   Parameters:
   json_arr: A json_arr_t onject.
   idx: The index from which the value is to be retrieved.

   Return: A json_t object at the idx, NULL if fail.

   If the operation fails because of out of bound array access, json_error is set to json_error_arr_out_of_bound_access.

5. **void json_arr_free(json_arr_t arr)**

   Free json_arr_t object.

   Parameters:
   arr: A json_arr_t object.

   Returns: void

- **Functions on json_obj_t**

1. **bool json_obj_insert(json_obj_t *json_obj_p, const char *key, json_t value)**

   Insert a json value to the json object.

   Parameters:
   json_obj_p: A pointer to a json_obt_t object
   key: key string
   value: A json value

   Returns: true if insertion success, false otherwise.

2. **json_t json_obj_get(const json_obj_t j_obj, const char *key)**

   Get a json value from json object with a key.

   Parameters:
   j_obj: A json object
   key: Respective key of the desired json value

   Returns: A the json_value if found, NULL otherwise

3. **void json_obj_free(json_obj_t j_obj)**

   Free the json object.

   Parameters:
   j_obj: A json object

   Returns: void

- **bool json_is_valid(const char *json_str)**

  This function will check whether a json string is valid or not. It check if the given string conforms to specification in https://tools.ietf.org/html/rfc4627

  Parameters:
  json_str: A json string.

  Return: true if valid, false if not valid

- **json_t json_parse(char *json_file_path)**

  This function will parse a json file into a json value.

  Parameters:
  json_file_path: Path to the json file.

  Returns: A json_t object on success, NULL if failed

  If the json file does not exist json_error is set to json_error_file_not_exit. If the json is invalid json_error is set to json_error_invalid_syntax. If parsing failed, json_error is set to json_error_parsing_failed.

- **char *read_json_file(const char *file_path)**

  Read a json file.

  Parameters:
  file_path: Path to the json file.

  Returns: A pointer to the json string. The json string is allocated by the function and must be free afterwards.

- **json_error**

  json_error will tell the cause of error if error occured during the processing of json. Following are the error types:
  - json_error_file_not_exist:
    josn file does not exist
  - json_error_invalid_syntax:
    invalid json syntax
  - json_error_parsing_failed:
    json parsing failed
  - json_error_dtype_mismatch_access:
    datatype mismatch access
  - json_error_arr_out_of_bound_access:
    array index out of bound access
  - json_error_hash_collision:
    hash collision while inserting value to json object
  - json_error_noerror:
    default value of json_error