Contribution Guidelines

=======================

Contributions to cJSON are welcome. If you find a bug or want to improve cJSON in another way, pull requests are appreciated.

For bigger changes, in order to avoid wasted effort, please open an issue to discuss the technical details before creating a pull request.

The further sections explain the process in more detail and provides some guidelines on how contributions should look like.

Branches

--------

There are two branches to be aware of, the `master` and the `develop` branch. The `master` branch is reserved for the latest release, so only make pull requests to the `master` branch for small bug- or security fixes (these are usually just a few lines). In all other cases, please make a pull request to the `develop` branch.

Coding Style

------------

The coding style has been discussed in [#24](https://github.com/DaveGamble/cJSON/issues/24). The basics are:

\* Use 4 spaces for indentation

\* No oneliners (conditions, loops, variable declarations ...)

\* Always use parenthesis for control structures

\* Don't implicitly rely on operator precedence, use round brackets in expressions. e.g. `(a > b) && (c < d)` instead of `a>b && c<d`

\* opening curly braces start in the next line

\* use spaces around operators

\* lines should not have trailing whitespace

\* use spaces between function parameters

\* use pronouncable variable names, not just a combination of letters

Example:

```c

/\* calculate the new length of the string in a printbuffer and update the offset \*/

static void update\_offset(printbuffer \* const buffer)

{

const unsigned char \*buffer\_pointer = NULL;

if ((buffer == NULL) || (buffer->buffer == NULL))

{

return;

}

buffer\_pointer = buffer->buffer + buffer->offset;

buffer->offset += strlen((const char\*)buffer\_pointer);

}

```

Unit Tests

----------

cJSON uses the [Unity](https://github.com/ThrowTheSwitch/Unity) library for unit tests. The tests are located in the `tests` directory. In order to add a new test, either add it to one of the existing files (if it fits) or add a new C file for the test. That new file has to be added to the list of tests in `tests/CMakeLists.txt`.

All new features have to be covered by unit tests.

Other Notes

-----------

\* Internal functions are to be declared static.

\* Wrap the return type of external function in the `CJSON\_PUBLIC` macro.