QGIS Documentation Guidelines

QGIS Project

2022 年 03 月 24 日

目次

QGIS ドキュメントは、米国太平洋時間の太平洋標準時の午前 0 時、午前 8 時、午後 4 時にサーバ上で自動作成されます。現在の状態は https://docs.qgis.org で入手できます。

QGIS ドキュメントのソースファイルは https://github.com/qgis/QGIS-Documentation で入手できます。これらは主に reStructuredText（reST）形式の構文を使用して記述され、HTML 出力を後処理するための Sphinx ツールセットからのスクリプトと組み合わされています。これらのツールの一般的な情報については http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html または https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html 参照。

以下の章では次のようなことを学ぶことができます。

- how to manage the documentation source files using git system and the GitHub platform on which they are stored
- ルールに準拠した方法でテキストを変更し、スクリーンショットを提供する方法
- あなたの行った変更を確実に公式ドキュメントに取り込み共有する方法

QGIS プロジェクトに貢献する方法に関する一般的な情報を探している場合は、 QGIS コミュニティに参加する にヘルプを見つけることができます。

第 1 章

貢献のための一歩一歩

- **より詳しく知りたい場合は**

注釈: ここでは手順の説明のために QGIS ドキュメントを使用しますが、以下で説明するコマンドおよび手順はすべて、QGIS ウェブサイトにも当てはまります。

あなたがこの文章を読まれているということはきっと、QGIS ドキュメントに貢献しようという気持ちがあって、そのための方法を探されているのでしょう。 ここで提供しようとしているのは、まさしくそれです。現在この文書では、目的を達成するための複数の方法を一通り案内し、従うべき主な手順を示し、使用可能な小技と知っておくべき落とし穴を教えています。

何か助けが必要なときには迷わず、修正しようとしている Issue レポートのコメントに書き込むか、 QGIS コミュニティチームメーリングリスト に投稿をしてください。より詳しくは ドキュメントを書く をご覧ください。

さあ、では集中して始めてみましょう。

Documentation sources are stored using the git version control system and are available on GitHub at https://github.com/qgis/QGIS-Documentation. A list of issues to fix and features to explain can be found at https://github.com/qgis/QGIS-Documentation/issues.

ちなみに: If you are a first-time contributor and do not know where to start from, you may be interested in tackling our welcoming reports.

There are two main ways, not mutually exclusive, to modify the files:

1. *GitHub* **のウェブインタフェースを使用する**
2. *Git* **コマンドラインツールを使用する**.

1.1 GitHub のウェブインタフェースを使用する

GitHub のウェブインタフェースでは次のことを行うことができます。

- ファイルを編集する
- 変更をプレビューし、コミットする
- 変更がメインリポジトリに挿入されるようにプルリクエストを行う
- ブランチを作成、更新または削除する

git や GitHub で使われる言葉にまだ馴染みがない場合は、GitHub の Hello-world プロジェクトを読んで、以下でも使われる基本的な語彙と操作について学んだ方がいいでしょう。

注釈: 報告された Issue をあなたが修正中なら

もしあなたが issue を修正するための変更を加えている最中なら、issue レポートにコメントをして、それをあなた自身に割り当ててください。これにより、複数の人が同じ issue に取り組むことを防ぐことができます。

1.1.1 Fork QGIS-Documentation

GitHub アカウント はすでに取得しているものと仮定すると、まず最初にするべきは、ドキュメントのソースファイルをフォークすることです。

QGIS-Documentation のリポジトリ ページに移動して、右上隅の 

ボタンをクリックします。

ご自身の GitHub アカウントに、QGIS-Documentation リポジトリ (`https://github.com/<YourName>/QGIS-Documentation`) が作られていることと思います。このリポジトリは公式の QGIS-Documentation リポジトリのコピーです。あなたに完全な書込権限が与えられていて、公式のドキュメントに影響を与えることなく変更を加えることができます。

1.1.2 Make changes

QGIS ドキュメンテーションに貢献するにはいくつかの異なった方法があります。以下ではそれらを別々に示しますが、あるプロセスから別のプロセスに切り替えることに何ら問題はありません。

選択肢 1： `Edit on GitHub` ショートカットを使用する

QGIS ドキュメントのそれぞれのページは、ページの右上にある `Edit on GitHub` というリンクをクリックすると、素早く簡単に編集することができます。

1. `Edit on GitHub` をクリックすると、`qgis:master` ブランチのファイルが開きます。ページの上部には、あなたにはこのリポジトリへの書込権限がないため、変更はあなたのリポジトリの新しいブランチで適用される旨を知らせるメッセージが、表示されます。
2. 変更を行います。ドキュメントは reStructureText シンタックスを使用して書かれていますので、変更の内容によっては、 **執筆のためのガイドライン** を参照しながら行う必要があるかもしれません。
3. 終了したら、行った変更について短いコメントを書いて、 *Propose changes* をクリックします。これによってあなたのリポジトリに新しい ブランチ (`patch-xxx`) が作成されます。
4. *Propose changes* をクリックすると、GitHub の *Comparing changes* ページに移動します。
   - すべての変更が終わったら、下の **プルリクエストで変更をシェアする** のセクションの、 **変更を比較** にスキップしてください。
   - QGIS に送信する前に追加の変更が必要な場合は、次の手順に従ってください。
     1. フォークしたあなたの QGIS-Documentation (`https://github.com/<YourName>/QGIS-Documentation`) リポジトリに移動します。

2. Branch: master をクリックして `` patch-xxx`` ブランチを探し、このブランチを選択します。 Branch: master ボタンが *Branch: patch-xxx* に変わります。
3. 下の **ファイルを修正する** に飛んでください。

注釈: `Edit on GitHub` ショートカットは左サイドバーの一番下のドロップダウンメニューからも利用できます。

選択肢 2：あなたのドキュメンテーションレポジトリに一時的な専用ブランチを作成する

あなたがフォークした QGIS-Documentation で直接ファイルを編集できます。

フォークした QGIS-Documentation リポジトリの左上隅にある Branch: master をクリックして、テキストフィールドに一意の名前を入力して新しい ブランチ を作成します。新しいブランチの名前は、修正しようとしている問題に関連していなければなりません。すると Branch: master ボタンは *Branch: branch_name* となるはずです。

ちなみに：変更はこの一時的な専用ブランチで行うこと。絶対に `master` ブランチでは行わないこと

`qgis/QGIS-Documentation` の `master` ブランチからあなたの QGIS-Documentation リポジトリに変更をマージする場合を除いて、 慣例として `master` ブランチでは変更を行わないでください。問題ごとに別々のブランチを使用することによって、他のブランチに干渉することなく、同時に複数の問題に取り組むことができます。間違えた場合は、いつでもブランチを削除して、master ブランチから新しいブランチを作成してやり直すことができます。

1.1.3 Modify files

1. フォークした QGIS-Documentation のソースファイルをブラウズして、修正する必要があるファイルに移動します。
2. **執筆のためのガイドライン** に従いながら修正を行います。
3. 終了したら、ページの一番下にある **Commit Changes** フレームに移動し、行った変更について短いコメントを書き、そして *Commit Changes* をクリックしてあなたのブランチに直接変更をコミットします。 *Commit directly to the branch_name branch.* が選択されていることを確認してください。
4. 問題を修正するために更新する必要がある他のファイルについて上記の手順を繰り返します。

1.1.4 Share your changes via Pull Request

あなたの変更を公式ドキュメントに統合するためにはプルリクエストをする必要があります。

注釈: `Edit on GitHub` **リンクを使用して始めた場合は**

変更をコミットした後、GitHub はあなたの `patch-xxx` ブランチで行った変更を `qgis/QGIS-Documentation` マスターブランチと比較する新しいページを自動的に開きます。

下の *Step 2* にスキップしてください。

Start a new pull request

QGIS-Documentation リポジトリのメインページに行き、 *New pull request* をクリックします。

Compare changes

一方が `base:master` 、もう一方が ``compare:branch_name` （図を参照）という 2 つのダイアログボックスが表示されている場合は、あなたの行った変更はあなたのリポジトリの中で、変更を加えたブランチからあなたのマスターブランチへとマージされるだけです。これを修正するために *compare across forks* と表示されているリンクをクリックします。

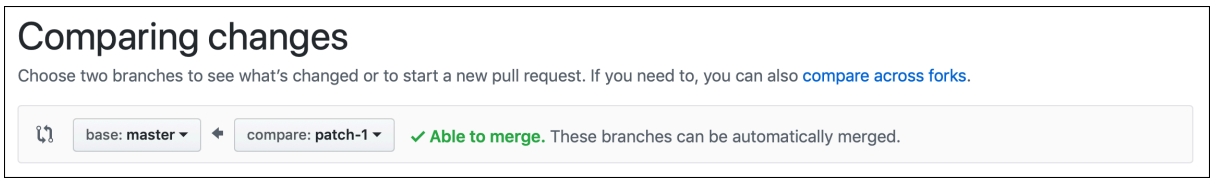


図 1.1 *Comparing changes* ページがこのようなものだったら、 *compare across forks* リンクをクリックします。

4 つのドロップダウンメニューが表示されていることと思います。このドロップダウンメニューによって、あなたのブランチで行った変更を、変更をマージしたい相手である公式のマスターブランチと比較することが可能になります。4 つのドロップダウンメニューは以下の通りです。

- **base fork** ：あなたの変更をマージしたい相手先のフォーク
- **base** ：あなたの変更をマージしたい base fork のブランチ
- **head fork** ：base fork に組み込みたい変更があるフォーク
- **compare** ： 変更が行われたブランチ

base fork で `qgis/QGIS-Documentation` を、base で `master` を選択します。head fork をあなたのリポジトリ `<YourName>/QGIS-Documentation` に、compare をあなたが変更を行ったブランチに設定します。

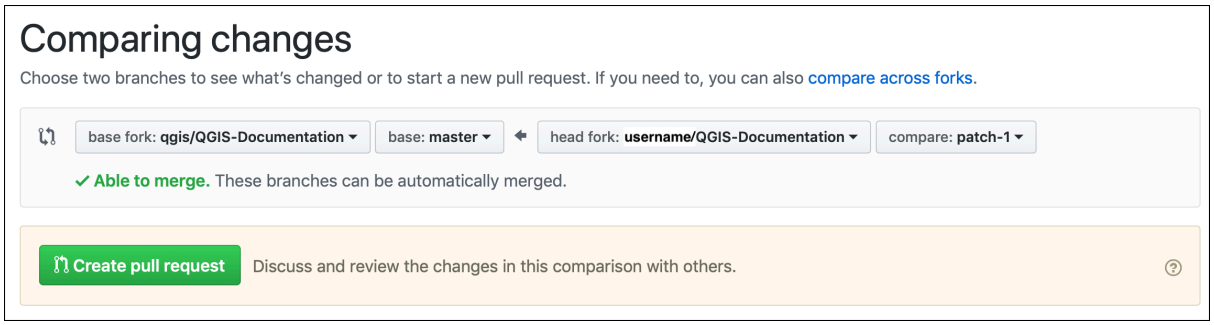


図 1.2　`qgis/QGIS-Documentation` とあなたのリポジトリとの間で変更を比較している

Able to merge と書いてある緑色のチェックマークは、あなたの変更が衝突することなく公式のドキュメントにマージできることを示しています。

Create pull request ボタンをクリックします。

警告: ✕ Can't automatically merge. **と表示されたら**

これは 競合 があることを意味します。他の誰かがあなたの変更と競合するコミットをしたので、あなたが修正中のファイルは対象のブランチの最新の状態を反映していません。それでもプルリクエストは作成できますが、マージを完了するには **競合** を修正する必要があります。

ちなみに: 最新バージョン の QGIS ドキュメントは、翻訳されてはいますが、保守は継続しており、問題が見つかれば修正が行われています。特定リリースの問題を修正する場合は、上記の手順で **base** を `master` から適切な `release_...` ブランチに変更してください。

Describe your pull request

テキストボックスが開きます。対処している問題に関するコメントを入力します。

これが特定の issue に関連する場合は、コメントに issue 番号を追加します。これは # と issue 番号 (**例** `#1234`) を入力することによって行われます。 直前に `fix` や `close` のような語が置かれている場合は、プルリクエストがマージされると、すぐにその issue はクローズされます。

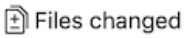
あなたが変更を行ったすべてのドキュメントのページへのリンクを含めてください。

Create pull request をクリックします。

Review and comment pull request

上で見たように、誰でもプルリクエストを通してドキュメントの修正を提案することができます。同様に、誰でも質問や コメント でプルリクエストをレビューすることが可能です。おそらくそれは、プロジェクトのガイドラインに沿っていない執筆スタイルだったり、重要な細部やスクリーンショットが変更に欠けていたり、あるいはすべてが素晴らしくて申し分なさそうだ、ということだったりするでしょう。レビューはドキュメントの形式と実質の両面において、貢献の質を高める助けになります。

プルリクエストをレビューするには

1. pull requests のページ へ移動して、コメントをしたいプルリクエストをクリックします。
2. ページの一番下に、このプルリクエストについて全体的なコメントを残すことのできるテキストボックスがあると思います。
3. 特定の行についてコメントをするには、
    1. 

をクリックして、コメントをしたいファイルを探します。 変更を確認するために *Display the source diff* をクリックしないといけないかもしれません。
    2. コメントしたい行までスクロールして、  をクリックします。テキストボックスが開きますので、コメントを残すことができます。

特定の行に対するコメントは、次のいずれかの方法で公開することができます。

- *Add single comment* ボタンを使って、単独のコメントとして公開する。この場合コメントはその都度即座に公開されます。コメント数が少ない場合や他のコメントにコメントする場合にのみ、この方法を使ってください。
- *Start a review* ボタンを押して、レビューの一部として公開する。コメントは検証後に自動的に送信されないため、後で編集やキャンセルしたり、レビューの主要な点を要約したりそのプルリクエストについての全体的な指示を追加したり、そのプルリクエストを承認するかどうかを述べたりすることができます。こちらの方がよりよい方法でしょう。より柔軟ですし、レビューを構造化したりコメントを編集したりして準備ができてから公開することができますし、リポジトリのフォロワーにコメントごとに通知を送るのではなく、ひとつだけ通知を送ることができるからです。 より詳しくは を参照してください。

図 1.3　特定の行に対して修正の提案とともにコメントを行う

行に対するコメントには提案を埋め込むことができ、プルリクエストの作成者はこれをプルリクエストに適用することができます。提案を追加するには、コメントテキストブロックの上にある Insert a suggestion ボタンをクリックして、提案ブロックの中のテキストを修正します。

ちなみに：プルリクエストへの提案をバッチとして追加したいときは

プルリクエストの作成者としてレビュワーからのフィードバックを直接プルリクエストに組み込むときに、対処すべき提案の数が多いためにそれらをバッチコミットとして追加したい場合は、コメントの一番下にある *Commit suggestion* ボタンを使用するのは避けてください。 すなわち、以下のようにします。

1. Files changed タブに移動します。
2. 取り込みたい提案のそれぞれで *Add suggestion to batch* を押します。押すごとにカウンターが増えるのが分かると思います。
3. 取り込みたいすべての提案をプルリクエストに適用する準備ができたら、どれでもいいので *Commit suggestions* ボタンを押して、この変更を説明するメッセージを入力します。

これによってすべての修正がひとつのコミットとしてブランチに追加されます。結果として、変更履歴が読みやすいものとなり、リポジトリのフォロワーに送信される通知の数も減ります。ついでに言えば、この処置によってあなたのクリック数も大きく減らすことができます。

Make corrections

新しいプルリクエストは自動的に プルリクエストリスト に追加されます。他の編集者や管理者があなたのプルリクエストを見直し、提案をしたり修正を求めたりするかもしれません。

A pull request will also trigger automated build checks (eg, for rst formatting, python code syntaxes), and reports are displayed at the bottom of the page. If an error is found, a red cross will appear next to your commit. Click on the red cross or on `Details` in the summary section at the bottom of the pull request page to see the details of the error. You'll have to fix any reported errors or warnings before your changes are committed to the `qgis/QGIS-Documentation` repository.

メインリポジトリにマージされるまでは、プルリクエストに修正を加えることができます。それはプルリクエストを改善するためだったり、提案された修正に対処するためだったり、あるいはビルドエラーを修正するためだったりするでしょう。

変更するにはプルリクエストのページで Files changed をクリックして、変更したいファイル名の横にある鉛筆ボタン をクリックします。

プルリクエストで使用したのと同じブランチで変更を加えた場合、追加された変更はすべてプルリクエストに自動的に追加されます。このため、追加の変更は、プルリクエストで修正しようとしている問題にその変更が関連する場合にのみ、行うようにしてください。

別の問題を解決したい場合は、それらの変更に対して新しいブランチを作成して上記のステップを繰り返します。

ビルドエラーが修正され、あなたと管理者が変更に満足したら、管理者はあなたの貢献をマージします。

1.1.5 Delete your merged branch

変更がマージされた後でブランチを削除できます。古いブランチを削除すると、未使用のブランチや古いブランチをリポジトリに保存しておく必要がなくなります。

1. フォークしたあなたの QGIS-Documentation リポジトリ (`https://github.com/<YourName>/QGIS-Documentation`) に移動します。
2. *Branches* タブをクリックします。 *Your branches* の下にあなたのブランチがあることと思います。
3. 不要なブランチの Delete this branch アイコンをクリックして削除します。

1.2 Git コマンドラインツールを使用する

GitHub のウェブインターフェイスは、簡単なやり方で QGIS-documentation レポジトリの更新に貢献することができますが、下記のためのツールは提供していません。

- 複数のコミットをまとめることによって変更履歴をきれいにする
- メインリポジトリとの間で生じうる衝突を解決する

- あなたの変更をテストするためにドキュメントをビルドする

より高度で強力なツールへアクセスし、リポジトリのコピーをローカル環境に持つためには、ハードドライブ上に git をインストール する必要があります。しばしば必要となる基本的な事項は以下で説明されています。そこではウェブインターフェイスを使用する場合であっても気をつけるべきルールを学ぶことができるでしょう。

以下のコードサンプルでは、`$` で始まる 行はあなたが入力すべきコマンドを示します。一方、`#` で始まる 行はコメントです。

1.2.1 ローカルリポジトリ

さあ、QGIS-Documentation リポジトリの**あなた用の**コピーをローカル環境に取得する準備はできましたね。

ウェブ URL を使った以下のコマンドで、QGIS ドキュメンテーションリポジトリを複製し取得することができます。

```
# move to the folder in which you intend to store the local repository
$ cd ~/Documents/Development/QGIS/
$ git clone https://github.com/<YourName>/QGIS-Documentation.git
```

The former command line is simply an example. You should adapt both the path and the repository URL, replacing `<YourName>` with your github user name.

次に以下の確認を行ってください。

```
# Enter the local repository
$ cd ./QGIS-Documentation
$ git remote -v
origin  https://github.com/<YourName>/QGIS-Documentation.git (fetch)
origin  https://github.com/<YourName>/QGIS-Documentation.git (push)
$ git branch
* master
```

- *origin* は、あなたの QGIS-Documentation リポジトリの、リモートリポジトリにつけられた名前です。
- *master* はデフォルトのメインブランチです。貢献する際にはこのブランチを使用してはいけません。**絶対に** です！

SSH プロトコルを使用して QGIS ドキュメンテーションリポジトリを複製することもできます。

```
# move to the folder in which you intend to store the local repository
$ cd ~/Documents/Development/QGIS/
$ git clone git@github.com:<YourName>/QGIS-Documentation.git
```

ちなみに: Permission denied (publickey) エラーが出た時は？

上記のコマンドで Permission denied (publickey) というエラーが出た場合は、おそらくはあなたの SSH key に問題があります。詳細は GitHub のヘルプ を参照してください。

SSH プロトコルを使用した場合は確認は以下のようになります。

```
# Enter the local repository
$ cd ./QGIS-Documentation
$ git remote -v
origin  git@github.com:<YourName>/QGIS-Documentation.git (fetch)
origin  git@github.com:<YourName>/QGIS-Documentation.git (push)
$ git branch
* master
```

これで始めることができますが、長い間のうちには、あなたの貢献をプッシュした際（GitHub のプロセスではプルリクエストと呼びます）に、たくさんの問題が生じると思います。これは公式の qgis/QGIS-Documentation リポジトリの master ブランチが、あなたのローカル／リモートリポジトリからだんだんと差異を増してずれていくことによるものです。このため、常にメインリモートリポジトリの状態を追跡したうえで、ブランチにおける作業を行う必要があります。

1.2.2 もうひとつのリモートリポジトリを追加

メインプロジェクトで行われた作業を追跡できるようにするために、ローカルリポジトリに新しいリモートリポジトリを追加します。この新しいリモートリポジトリは、QGIS プロジェクト公式の QGIS-Documentation リポジトリです。

```
$ git remote add upstream https://github.com/qgis/QGIS-Documentation.git
$ git remote -v
origin  https://github.com/<YourName>/QGIS-Documentation.git (fetch)
origin  https://github.com/<YourName>/QGIS-Documentation.git (push)
upstream        https://github.com/qgis/QGIS-Documentation.git (fetch)
upstream        https://github.com/qgis/QGIS-Documentation.git (push)
```

ローカルリポジトリにリモートリポジトリを追加するときにも、同様に SSH プロトコルを使うことができます。

```
$ git remote add upstream git@github.com:qgis/QGIS-Documentation.git
$ git remote -v
origin  git@github.com:<YourName>/QGIS-Documentation.git (fetch)
origin  git@github.com:<YourName>/QGIS-Documentation.git (push)
upstream        git@github.com:qgis/QGIS-Documentation.git (fetch)
upstream        git@github.com:qgis/QGIS-Documentation.git (push)
```

これで 2 つのリモートリポジトリの、どちらかを選ぶことができるようになりました。

- *origin* は **あなたの** リモートリポジトリに、あなたのローカルブランチをプッシュするときに使用します。
- *upstream* は、あなたの貢献を公式のリポジトリにマージしたり（その権限がある場合）、あなたのロー

カルリポジトリのマスターブランチを公式リポジトリのマスターブランチに従って更新したりする際に使用します。

注釈: *upstream* は標準の名前のようになっていますが、実際はただのラベルですので、あなたの好きなように名前をつけることができます。

1.2.3 ベースブランチを更新する

新しい貢献に取り組む前には、必ず、ローカルリポジトリのマスターリポジトリをアップデートしなければなりません。

```
# switch to master branch (it is easy to forget this step!)
$ git checkout master
# get "information" from the master branch in the upstream repository
# (aka qgis/QGIS-Documentation's repository)
$ git fetch upstream master
# merge update from upstream/master to the current local branch
# (which should be master, see step 1)
$ git merge upstream/master
# update **your** remote repository (aka <YourName>/QGIS-Documentation)
$ git push origin master
```

これであなたのローカルリポジトリとリモートリポジトリ双方の `master` ブランチを、公式 QGIS-Documentation リポジトリの `master` ブランチに一致するようアップデートしましたので、貢献の作業を始めることができます。

注釈: リリースドキュメントに貢献したい場合はブランチを移動すること

testing 版ドキュメントとともに、latest release 版ドキュメントについても問題を修正する作業が続けられていますので、このドキュメントに対して貢献することも可能です。コード中の `master` を最新の対応するブランチに置き換えた上で、前セクションのサンプルコードに従ってください。

1.2.4 制作ブランチに取り組む

ベースブランチを最新に更新しましたので、次は自分の貢献を追加する専用のブランチを作成する必要があります。作業は必ずベースブランチ以外のブランチで行ってください！これは常にです！

```
# Create a new branch
$ git checkout -b myNewBranch
# checkout means go to the branch
# and -b flag creates a new branch if needed, based on current branch
# Let's check the list of existing branches (* indicates the current branch)
$ git branch
```

(次のページに続く)

(前のページからの続き)

```
master
release_2.18
...
* myNewBranch
# You can now add your contribution, by editing the concerned file(s)
# with any application (in this case, vim is used)
$ vim myFile
# once done
$ git add myFile
$ git commit
```

commit/push コマンドについて一言：

- ひとつの貢献（それ以上分割することが不可能な変更）だけをコミットするようにしてください。すなわち一度にひとつの問題だけに取り組んでください。
- コミットのタイトルおよび説明文の中で、変更の内容を丁寧に説明するようにしてください。最初の行はタイトルです。大文字で始め、80 文字以内に収め、最後には . を付けないでください。簡潔にしてください。コミットの説明文は長くなってもよいので、より多く詳細について語ることができます。最後は . で終了してください。
- Issue を参照するには、 `#` を頭につけた Issue 番号を使用してください。チケットを修正する場合は `Fix` をその前につけておくと、コミットによってチケットが閉じられます。

さあ変更が保存されローカルブランチにコミットされました。プルリクエストを作成するためには、リモートリポジトリに送信する必要があります。

```
$ git push origin myNewBranch
```

1.2.5 変更を共有する

これであなたの github のリポジトリに行って、前のセクションで触れたように **プルリクエストを作成** することができます。作成したプルリクエストが、自分のブランチから公式の QGIS-Documentation リポジトリ中のターゲットとしているリモートブランチへものであることを確認してください。

1.2.6 ローカルおよびリモートリポジトリをクリーンアップ

プルリクエストが公式の QGIS-Documentation にマージされたら、あなたの制作ブランチは削除してかまいません。この方法で多くの作業をこなした場合、数週間のうちに用済みのブランチがたくさんできると思います。ですので以下のようにしてあなたのリポジトリをきれいに保ちましょう。

```
# delete local branch
$ git branch -d myNewBranch
# Remove your remote myNewBranch by pushing nothing to it
$ git push origin :myNewBranch
```

またローカルリポジトリ中の `master` ブランチを更新して最新の状態を保つことも忘れないでください！

1.3 より詳しく知りたい場合は

- 上記の Github ウェブインターフェイスと git コマンドラインツール以外にも、ドキュメントへの貢献を作成および管理するために使用できる GUI アプリケーション があります。
- プルリクエストの変更が、ターゲットブランチにプッシュされた最近の変更と競合している場合は、マージが可能になるよう、先にこの競合を解決する必要があります。
  - 競合が競合する数行に関連している場合は、Github のプルリクエストのページに *Resolve conflicts* ボタンがあります。 https://help.github.com/articles/resolving-a-merge-conflict-on-github/ で説明されているようにボタンを押し、問題を解決して下さい
  - 競合がファイルの名前変更または削除を伴う場合は、git コマンドラインを使用して競合を解決する必要があります。典型的には、最初に `git rebase targetBranch` 呼び出しを使ってターゲットブランチの上にあなたのブランチをリベースし、報告された衝突を修正しなければなりません。詳細は https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line/ をご覧ください。
- 場合によっては、校正プロセスの最後に、変更が複数のコミットに分割されてしまうことがあります。分割されたコミットが必ずしもそれだけの価値があるわけではありません。 Git コマンドラインは、これらのコミットをより少数の、より意味のあるコミットメッセージに変換するのに役立ちます。より詳しくは https://help.github.com/articles/using-git-rebase-on-the-command-line/ を参照してください。

第 2 章

執筆のためのガイドライン

総じて、QGIS プロジェクトのために reST ドキュメントを作成するときには、 Python documentation style guidelines に従ってください。簡便のために以下に、QGIS ドキュメントを書く際に依拠すべき一般的なルー

ルを、一通り示します。

2.1 ドキュメントを書く

2.1.1 見出し

ドキュメントのそれぞれのウェブページには、ひとつの `.rst` ファイルが対応しています。

テキストを構造化するために使用されるセクションはそれらのタイトルを通じて識別されます。タイトルには下線（及び第 1 レベルに対して上線）が引かれます。同じレベルのタイトルは下線装飾のために同じ文字を使用する必要があります。QGIS 文書では、章、セクション、サブセクションと minisec に対して以下のスタイルを使用する必要があります。

```
********
Chapter
********

Section
=======

Subsection
----------

Minisec
.......

Subminisec
^^^^^^^^^^
```

2.1.2 リスト

リストはテキストを構造化するのに役立ちます。こちらはすべてのリストに共通な簡単な規則のいくつかです：

- すべての項目を大文字で始めてください
- 単一の単文のみを含むリスト項目の後に句読点を使用しないでください
- 複数の文または 1 つの複合文からなるリスト項目の句読点としてピリオド（ . ）を使用します

2.1.3 行内タグ

You can use tags to emphasize items.

- **メニュー GUI** ：サブメニューを選択したり、特定の操作、またはこのような配列の何らかの部分配列を選択するなど、メニュー選択の完全な配列をマークします。

```
:menuselection:`menu --> submenu`
```

- **Dialogs and Tab titles**: Labels presented as part of an interactive user interface including window titles, tab titles, button and option labels.

```
:guilabel:`title`
```

- **Filenames and directories**

```
:file:`README.rst`
```

- **Icons with popup text**

```
|icon| :sup:`popup_text`
```

（以下の **画像** を参照してください）。

- **キーボードショートカット**

```
:kbd:`Ctrl+B`
```

will show `Ctrl+B`

キーボードショートカットを説明するときは、次の規則を使用してください：

  - Letter keys are displayed using uppercase: `S`
  - Special keys are displayed with an uppercase first letter: `Esc`
  - キーの組み合わせはキー同士の間に空白を入れずに + 記号を表示し、 `Shift+R` のように表示します。

- **ユーザーテキスト**

```
``label``
```

2.1.4 Labels/references

Anchors inside the text can be used to create hyperlinks to sections or pages.

以下の例は、セクション（例えば、ラベル/参照タイトル）のアンカーを作成します

```
.. _my_anchor:

Label/reference
---------------
```

同じページ 中の参照を呼び出すために使用するのは

```
see my_anchor_ for more information.
```

これが返すのは：

詳細については *my_anchor* を参照してください。

Notice that it will jump to the line/thing following the 'anchor'. You do not need to use apostrophes, but you do need to have empty lines after the anchor.

文書内のどこからでも 同じ場所にジャンプする別の方法は、 `:ref:` 役割を使用することです。

```
see :ref:`my_anchor` for more information.
```

which will create a link with the caption instead (in this case the title of this section!):

詳細については *Labels/references* を参照。

So, reference 1 (*my_anchor*) and reference 2 (*Labels/references*). Because the reference often displays a full caption, it is not really necessary to use the word *section*. Note that you can also use a custom caption to describe the reference:

```
see :ref:`Label and reference <my_anchor>` for more information.
```

which returns:

詳細については **ラベルや参照** を参照。

2.1.5 図と画像

画像

画像を挿入するには、使用します

```
.. figure:: /static/common/logo.png
   :width: 10 em
```

これを返します

置換

テキスト内には画像を置くか、どこでも使用される別名を追加できます。段落内で画像を使用するには、 最初に `source/substitutions.txt` ファイル中に別名を作成します：

```
.. |nice_logo| image:: /static/common/logo.png
               :width: 1 em
```

and then call it in your paragraph:

```
My paragraph begins here with a nice logo |nice_logo|.
```

This is how the example will be displayed:

My paragraph begins here with a nice logo  .

To allow preview rendering in GitHub that is as close as possible to HTML rendering, you will also need to add the image replacement call at the end of the file you changed. This can be done by copy-pasting it from `substitutions.txt` or by executing the `scripts/find_set_subst.py` script.

注釈: Currently, to ensure consistency and help in the use of QGIS icons, a list of aliases is built and available in the **置換参照と定義** chapter.

図

```
.. _figure_logo:

.. figure:: /static/common/logo.png
   :width: 20 em
   :align: center

   A caption: A logo I like
```

結果は以下のようになります。

図 2.1　キャプション：私の好きなロゴ

To avoid conflicts with other references, always begin figure anchors with `_figure_` and use terms that easily connect to the figure caption. While only the centered alignment is mandatory for the image, feel free to use any other options for figures (such as `width`, `height`, `scale`...) if needed.

The scripts will insert an automatically generated number before the caption of the figure in the generated HTML and PDF versions of the documentation.

キャプションを使用するには（ *My caption* を参照)、図ブロックに空白行の後ろに字下げテキストを挿入します。

A figure can be referenced using the reference label like this:

```
see :numref:`figure_logo`
```

renders like this:

see 図 2.1

This is the preferred way of referencing figures.

注釈: For `:numref:` to work, the figure **must have a caption**.

Avoid using `:ref:` instead of `:numref:` for reference,since this returns the full caption of the image.

```
see :ref:`figure_logo`
```

renders like this:

キャプション：私の好きなロゴ 参照

表

A simple table can be coded like this

```
=======  =======  =======
x        y        z
=======  =======  =======
1        2        3
4                 5
=======  =======  =======
```

It will render like this:

x	y	z
1	2	3
4		5

Use a \ (backslash) followed by an empty space to leave an empty space.

You can also make more complicated tables and reference them:

```
.. _my_drawn_table:

+---------------+--------------------+
| Windows       | macOS              |
+---------------+--------------------+
| |win|         | |osx|              |
+---------------+--------------------+
| and of course not to forget |nix|  |
+-------------------------------------+

My drawn table, mind you this is unfortunately not regarded as a caption

You can reference it like this: my_drawn_table_.
```

結果：

ウィンドウズ	macOS
	
そしてもちろん  を忘れないように	

My drawn table, mind you this is unfortunately not regarded as a caption

my_drawn_table のように参照できます。

For even more complex tables, it is easier to use `list-table`:

```
.. list-table::
   :header-rows: 1
```

(次のページに続く)

(前のページからの続き)

```
   :widths: 20 20 20 40

   * - What
     - Purpose
     - Key word
     - Description
   * - **Test**
     - ``Useful test``
     - complexity
     - Geometry.  One of:

       * Point
       * Line
```

結果：

What	目的	Key word	説明
Test	`Useful test`	complexity	Geometry. One of: • 点 • Line

2.1.6 索引

索引は読者がドキュメント内で必要な情報を見つけるのを手助けする手軽な方法です。QGIS ドキュメントは必須の索引項目のみを提供しています。本当に有用な（よく整理され、首尾一貫し、相互に関連づけられた）索引項目のセットのみを提供する助けとなるルールがいくつかあります。

- 索引は、人間が読めるもの、理解できるもの、翻訳可能なものでなければなりません。索引は複数の単語から作ることができますが、それらを繋ぐ不要な _ や - などは使用しないようにしてください。たとえば `loading_layers` や `loadingLayers` ではなく、 `Loading layers` としてください。
- 特別な綴りを持つ単語でない限りは、索引語の最初の文字のみを大文字にします。たとえば、`Loading layers`、 `Atlas generation`、 `WMS`、 `pgsql2shp` などとします。
- 常に現在の 索引語リスト に注意を払うことで、より好適な表現を正しい綴りで再利用し、不要な重複は避けるようにしてください。

RST には索引用のタグがあります。行内タグ `:index:` は通常のテキスト中で次のように使います。

```
QGIS can load several :index:`Vector formats` supported by GDAL/OGR ...
```

ブロックレベルのマークアップの `.. index::` を使うこともできます。これは次の段落の始まりにリンクします。上記ルールにより、ブロックレベルタグの使用を推奨します。

```
.. index:: WMS, WFS, Loading layers
```

single 、 pair 、 see のような索引パラメータの使用も推奨します。これはより構造化され相互に関連した索引語テーブルを作るのに有用です。索引作成についてのより詳しい情報は Index generating を参照してください。

2.1.7 特別なコメント

Sometimes, you may want to emphasize some points of the description, either to warn, remind or give some hints to the user. In QGIS Documentation, we use reST special directives such as `.. warning::`, `.. seealso::`, `.. note::` and `.. tip::`. These directives generate frames that highlight your comments. See Paragraph Level markup for more information. A clear and appropriate title is required for both warnings and tips.

```
.. tip:: **Always use a meaningful title for tips**

   Begin tips with a title that summarizes what it is about. This helps
   users to quickly overview the message you want to give them, and
   decide on its relevance.
```

2.1.8 短いコード

You may also want to give examples and insert code snippets. In this case, write the comment below a line with the `::` directive inserted. For a better rendering, especially to apply color highlighting to code according to its language, use the code-block directive, e.g. `.. code-block:: xml`. More details at Showing code.

注釈: While texts in note, tip and warning frames are translatable, be aware that code block frames do not allow translation. So avoid comments not related to the code and keep comments as short as possible.

2.1.9 脚注

Please note: Footnotes are not recognized by any translation software and it is also not converted to pdf format properly. So, if possible, don't use footnotes within any documentation.

これは、脚注を作成するためのものです（例として示します[*1] ）

```
blabla [1]_
```

これが指しているのは：

[*1] コアプラグインの更新