Test the Web Forward

Github Test Submission

All the basics that you need to know are documented on this page, but for the full GitHub documentation, visit help.github.com.

Setup

1. Create a GitHub account if you do not already have one on github.com
2. Download and install the latest version of Git: http://git-scm.com/downloads. Please refer to the instruction there for different platforms.
3. Configure your settings so your commits are properly labeled:

   On Mac or Linux or Solaris, open the Terminal.

   On Windows, open Git Bash (From the Start Menu > Git > Git Bash).

   At the prompt, type:

   ```
   $ git config --global user.name "Your Name"
   ```

   This will be the name that is displayed with your test submissions

   Next, type:

   ```
   $ git config --global user.email "your_email@address.com"
   ```

   This should be the email address you used to create the account in Step 1.

   Next, type:

   ```
   $ git config --global push.default upstream
   ```

   This ensures that git push will never unintentionally create or update a remote branch.

4. (Optional) If you don't want to enter your username and password every time you talk to the remote server, you'll need to set up password caching. See Mac Instructions, or Windows Instructions, or Linux Instructions. *On all of those pages, see 'Password caching'.*

Test Repositories

The test repository that you contribute to will depend on the specification that you are testing. Currently there are two test repositories, one for CSS specification tests and the main W3C repository that contains tests for all other specificatons:

Main W3C test repository: github.com/w3c/web-platform-tests

CSS specification test repository: github.com/w3c/csswg-test

Fork

Now that you have Git set up, you will need to fork the test repository. This will enable you to submit your tests using a pull request (more on this below).

1. In the browser, go the the GitHub page for the test repository:

   CSS test repository: github.com/w3c/csswg-test

   Main W3C test repository: github.com/w3c/web-platform-tests

2. Click the Fork button in the upper right.

3. The fork will take several seconds, then you will be redirected to your GitHub page for this forked repository. If you forked the HTML test repo (for example), you will now be at **https://github.com/username/web-platform-tests**.

4. After the fork is complete, you're ready to clone.

Clone

If your fork was successful, the next step is to clone (download a copy of the files).

Clone the test repo

At the command prompt, cd into the directory where you want to keep the tests.

- If you forked the W3C Web Platform tests:

  ```
  $ git clone --recursive https://github.com/username/web-platform-tests.git
  ```

  If you forked the CSS tests:

  ```
  $ git clone --recursive https://github.com/username/csswg-test.git
  ```

  This will download the tests into a directory named for the repo: `./web-platform-tests` *or* `./csswg-test`.

- You should now have a full copy of the test repository on your local machine. Feel free to browse the directories on your hard drive. You can also browse them on github.com and see the full history of contributions there.

Clone the submodules

- If you cloned the test repo and used the `--recursive` option, you'll find its submodules in `[repo-root]/resources/`.
- If you cloned the the test repo and did not use the `--recursive` option, you will likely have an empty `resources` directory at the root of your cloned repo. You can clone the submodules with these additional steps:

```
$ cd test-repo-root
$ git submodule init
$ git submodule update
```

  You should now see the submodules in the repository. For example, `testharness` *files in should be in the resources directory.*

Configure Remote / Upstream

Synchronizing your forked repository with the W3C repository will enable you to keep your forked local copy up-to-date with the latest commits in the W3C repository.

1. On the command line, navigate to to the directory where your forked copy of the repository is located.
2. Make sure that you are on the master branch. This will be the case if you just forked, otherwise switch to master.

```
$ git checkout master
```

3. Next, add the remote of the repository your forked. This assigns the original repository to a remote called "upstream"

   If you forked the Web Platform Tests repository:

```
$ git remote add upstream https://github.com/w3c/web-platform-tests.git
```

   If you forked the CSSWG-test repository:

```
$ git remote add upstream https://github.com/w3c/csswg-test.git
```

4. To pull in changes in the original repository that are not present in your local repository first fetch them:

```
$ git fetch upstream
```

Then merge them into your local repository:

```
$ git merge upstream/master
```

For additional information, please see the GitHub docs.

Branch

Now that you have everything locally, create a branch for your tests.

Note: If you have already been through these steps and created a branch and now want to create another branch, you should always do so from the master branch. To do this follow the steps from the beginning of the previous section. If you don't start with a clean master branch you will end up with a big nested mess.

At the command line:

```
$ git checkout -b username/topic
```

This will create a branch named `username/topic` and immediately switch this to be your active working branch.

The branch name should describe specifically what you are testing.
For Example:

```
$ git checkout -b username/flexbox-flex-direction-prop
```

You're ready to start writing tests! Come back to this page you're ready to commit them or submit them for review.

Commit

Before you submit your tests for review and contribution to the main test repo, you'll need to first commit them locally, where you now have your own personal version control system with git. In fact, as you are writing your tests, you may want to save versions of your work as you go before you submit them to be reviewed and merged.

1. When you're ready to save a version of your work, go to the command prompt and cd to the directory where your files are.
2. First, ask git what new or modified files you have:

```
$ git status
```

This will show you files that have been added or modified.

3. For all new or modified files, you need to tell git to add them to the list of things you'd like to commit:

```
$ git add [file1] [file2] ... [fileN]
```

Or:

```
$ git add [directory_of_files]
```

4. Run `git status` again to see what you have on the 'Changes to be committed' list. These files are now 'staged'.
5. Alternatively, you can run `git diff --staged`, which will show you the diff of things to be committed.
6. Once you've added everything, you can commit and add a message to this set of changes:

```
$ git commit -m "Tests for indexed getters in the HTMLExampleInterface"
```

7. Repeat these steps as many times as you'd like before you submit.

Submit

If you're here now looking for more instructions, that means you've written some awesome tests and are ready to submit them. Congratulations and welcome back!

1. The first thing you do before submitting them to the W3C repo is to push them back up to the server.

   At the command prompt:

```
$ git push https://github.com/username/web-platform-tests.git username/topic
```

   Or, for short:

```
$ git push origin username/topic
```

   Note: Here, `origin` *refers to remote repo from which you cloned (downloaded) the files after you forked, referred to as web-platform-tests.git in the previous example;* `username/topic` *refers to the name of your local branch that you want to push.*

2. Now you can send a message that you have changes or additions you'd like to be reviewed and merged into the main (original) test repository. You do this by using a pull request. In a browser, open the GitHub page for your forked repository: **https://github.com/username/web-**

platform-tests.

3. Now create the pull request. There are several ways to create a PR in the GitHub UI. Below is one method and others can be found on GitHub.com

   a. Click the Pull Requests link on the right side of the UI, then click the New pull request button.

   b. On the left, you should see the base repo is the w3c/web-platform-tests. On the right, you should see your fork of that repo. In the branch menu of your forked repo, switch to `username/topic` **Note:** If you see *'There isn't anything to compare'*, click the Edit button and make sure your fork and your `username/topic` branch is selected on the right side.

   c. Select the Click to create a pull request for this comparison link at the top.

   d. Scroll down and review the diff

   e. Scroll back up and in the Title field, enter a brief description for your submission.

   Example: "Tests for CSS Transforms skew() function."

   f. If you'd like to add more detailed comments, use the comment field below.

   g. Click Send pull request

4. Wait for feedback on your pull request and once your pull request is accepted, detele youre branch (see ' When Pull Request is Accepted').

That's it! If you're currently at a Test the Web Forward event, find an expert nearby and ask for a review. If you're doing this on your own (AWESOME!), your pull request will be go into a queue and will be reviewed soon.

Modify

Once you submit your pull request, a reviewer will check your proposed changes for correctness and style. It is likely that this process will lead to some comments asking for modifications to your code. When you are ready to make the changes, follow these steps:

1. Check out the branch corresponding to your changes e.g. if your branch was called `username/topic` run:

```
$ git checkout username/topic
```

2. Make the changes needed to address the comments, and commit them just like before.

3. Push the changes to the remote branch containing the pull request:

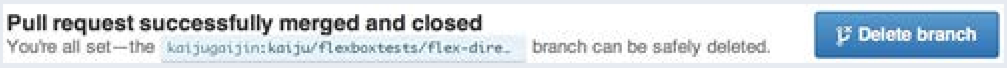
```
$ git push origin username/topic
```

4. The pull request will automatically be updated with the new commit. Note for advanced users: it is generally discouraged to rebase your pull request before review is complete. Tests typically have few conflicts so this should not be a problem in the common case.

Sometimes it takes multiple iterations through a review before the changes are finally accepted. Don't worry about this; it's totally normal. The goal of test review is to work together to create the best possible set of tests for the web platform.

Cleanup

Once your pull request has been accepted, you will be notified in the GitHub UI and you may get an email. At this point, your changes have been merged into the main test repository. You do not need to take any further action on the test but you should delete your branch. This can easily be done in the GitHub UI by navigating to the pull requests and clicking the 'Delete Branch' button.


Pull request successfully merged and closed
You're all set—the kaijugoijin:kaiju/flexboxtests/flex-dire… branch can be safely deleted.

Alternatively, you can delete the branch on the command line.

```
$ git push origin --delete <branchName>
```

Tips & Tricks

The following workflow is recommended:

1. Start branch based on latest w3c/master
2. Write tests
3. Rebase onto latest w3c/master
4. Submit tests
5. Stop fiddling with the branch base until review is done
6. After the PR has been accepted, delete the branch. (Every new PR should come from a new branch.)
7. Synchronize your fork with the W3C repository by fetching your upstream and merging it. (See 'Configure Remote / Upstream')

You need to be able to set up remote upstream, etc. Please refer to Pro Git Book and enjoy reading.

Edit on GitHub »

Table of Contents

Unanswered questions?

Discuss | Contribute | FAQ | Terms | Sponsors