

Working with non-code files

In this article

Rendering and diffing images

3D File Viewer

Rendering CSV and TSV data

Rendering PDF documents

Rendering differences in prose documents

Mapping GeoJSON/TopoJSON files on GitHub

Working with Jupyter Notebook files on GitHub

Displaying Mermaid files on GitHub

GitHub Enterprise Server supports rendering and diffing in a number of non-code file formats.

Rendering and diffing images

GitHub Enterprise Server can display several common image formats, including PNG, JPG, GIF, PSD, and SVG. In addition to simply displaying them, there are several ways to compare differences between versions of those image formats.

Note:

- GitHub does not support comparing the differences between PSD files.
- If you are using the Firefox browser, SVGs on GitHub may not render.

Viewing images

You can directly browse and view images in your repository on your GitHub Enterprise Server instance.

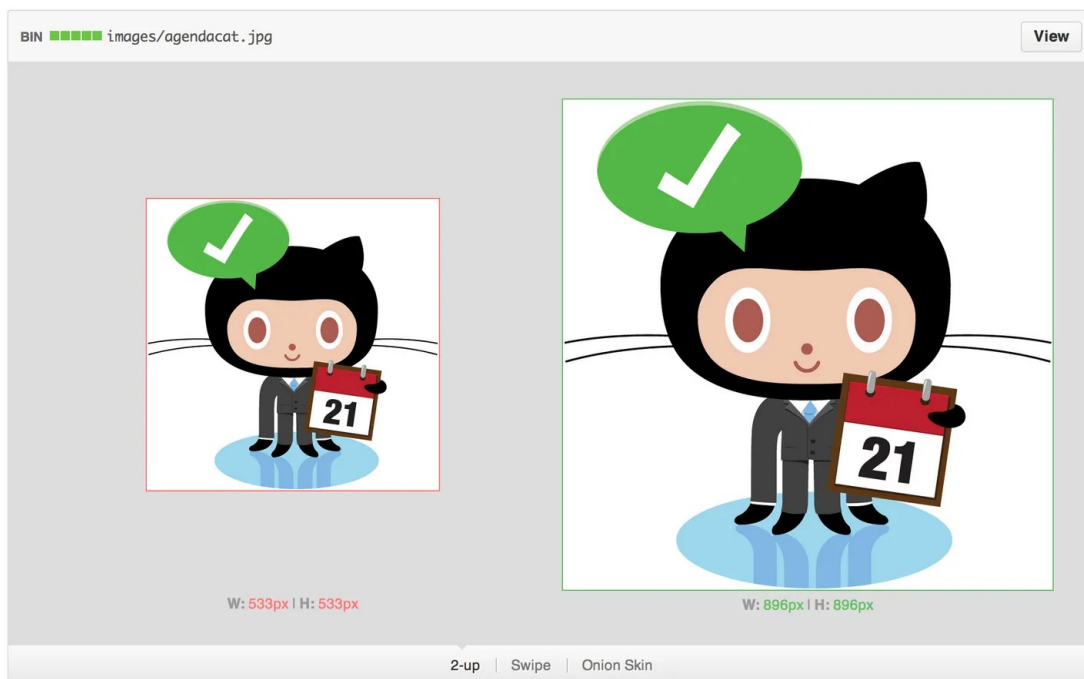
SVGs don't currently support inline scripting or animation.

Viewing differences

You can visually compare images in three different modes: [2-up](#), [swipe](#), and [onion skin](#).

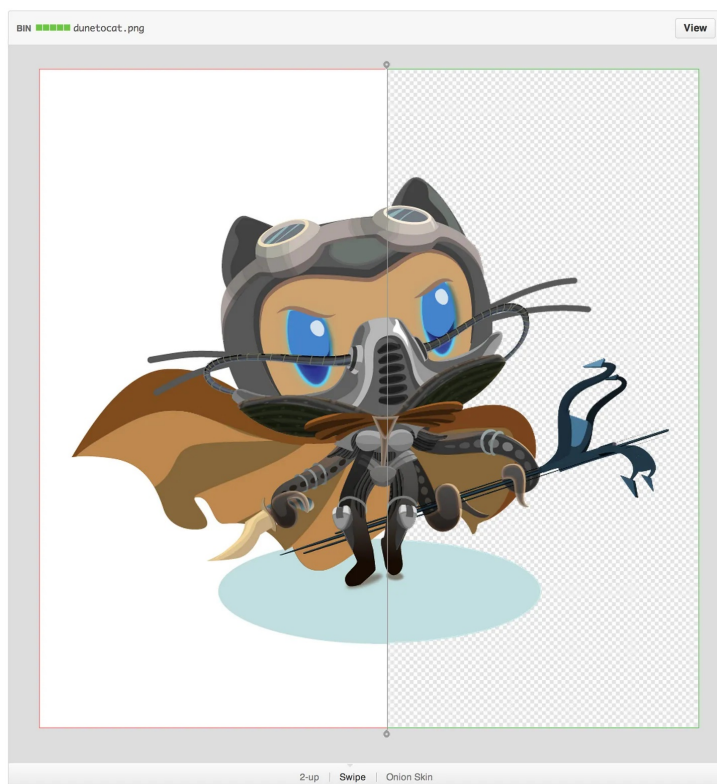
2-up

2-up is the default mode; it gives you a quick glimpse of both images. In addition, if the image has changed size between versions, the actual dimension change is displayed. This should make it very apparent when things are resized, such as when assets are upgraded to higher resolutions.



Swipe [🔗](#)

Swipe lets you view portions of your image side by side. Not sure if colors shifted between different versions? Drag the swipe slider over the area in question and compare the pixels for yourself.



Onion skin [🔗](#)

Onion Skin really comes in handy when elements move around by small, hard to notice amounts. Did an icon shift two pixels to the left? Drag the opacity slider back a bit and notice if things move around.

3D File Viewer [🔗](#)

GitHub Enterprise Server can host and render 3D files with the `.st/` extension.

When looking directly at an STL file on GitHub Enterprise Server you can:

- Click and drag to spin the model.
- Right click and drag to translate the view.
- Scroll to zoom in and out.
- Click the different view modes to change the view.

Fixing slow performance [↗](#)

If you see ⓘ in the corner of the viewer, with the tooltip "WebGL powered hardware support not available," then the WebGL technology is not available on your browser.

WebGL is necessary to take advantage of your computer's hardware to its fullest. We recommend you try browsers like [Chrome](#) or [Firefox](#), which ship with WebGL enabled.

Error: "Unable to display" [↗](#)

If your model is invalid, GitHub may not be able to display the file. In addition, files that are larger than 10 MB are too big for GitHub to display.

Embedding your model elsewhere [↗](#)

To display your 3D file elsewhere on the internet, modify this template and place it on any HTML page that supports JavaScript:

```
<script
src="https://embed.github.com/view/3d/<username>/<repo>/<ref>/<path_to_file>">
</script>
```

For example, if your model's URL is github.com/skalnik/secret-bear-clip/blob/master/stl/clip.stl, your embed code would be:

```
<script src="https://embed.github.com/view/3d/skalnik/secret-bear-clip/master/stl/clip.stl"></script>
```

By default, the embedded renderer is 420 pixels wide by 620 pixels high, but you can customize the output by passing height and width variables as parameters at the end of the URL, such as `?height=300&width=500`.

Note: `ref` can be a branch or the hash to an individual commit (like `2391ae`).

Rendering in Markdown [↗](#)

You can embed ASCII STL syntax directly in Markdown. For more information, see "[Creating diagrams](#)."

Rendering CSV and TSV data [↗](#)

GitHub supports rendering tabular data in the form of `.csv` (comma-separated) and `.tsv` (tab-separated) files.

915 lines (914 sloc) | 193.208 kb

RawBlameHistory

Q Search this file...

1	Title	Release Year	Locations
2	180	2011	555 Market St.
3	180	2011	Epic Roasthouse (399 Embarcadero)
4	180	2011	Mason & California Streets (Nob Hill)
5	180	2011	Justin Herman Plaza
6	180	2011	200 block Market Street
7	180	2011	City Hall
8	180	2011	Polk & Larkin Streets
9	180	2011	Randall Musuem
10	24 Hours on Craigslist	2005	
11	48 Hours	1982	
12	50 First Dates	2004	Rainforest Café (145 Jefferson Street)
13	A Jitney Elopement	1915	Golden Gate Park

When viewed, any `.csv` or `.tsv` file committed to a repository on your GitHub Enterprise Server instance automatically renders as an interactive table, complete with headers and row numbering. By default, we'll always assume the first row is your header row.

You can link to a particular row by clicking the row number, or select multiple rows by holding down the shift key. Just copy the URL and send it to a friend.

Searching data 🔗

If you want to find a certain value in your dataset, you can start typing in the search bar directly above the file. The rows will filter automatically.

Handling errors 🔗

Occasionally, you may discover that your CSV or TSV file isn't rendering. In those instances, a message appears above your raw text, suggesting what the error may be.

6 lines (6 sloc) | 385 Bytes

We can make this file [beautiful and searchable](#) if this error is corrected: No commas found in this CSV file in line 0.

1	Username; Identifier;One-time password;Recovery code;First name;Last name;Department;Location
2	booker12;9012;12se74;rb9012;Rachel;Booker;Sales;Manchester

Common errors include:

- Mismatched column counts. You must have the same number of separators in each row, even if the cell is blank
- Exceeding the file size. Our rendering only works for files up to 512KB. Anything bigger than that slows down the browser.
- Using unsupported delimiters, such as semicolons instead of commas.

Rendering PDF documents 🔗

GitHub supports rendering of PDF documents.

Currently, links within PDFs are ignored.

Rendering differences in prose documents 🔗


Commits and pull requests that include prose documents have the ability to represent

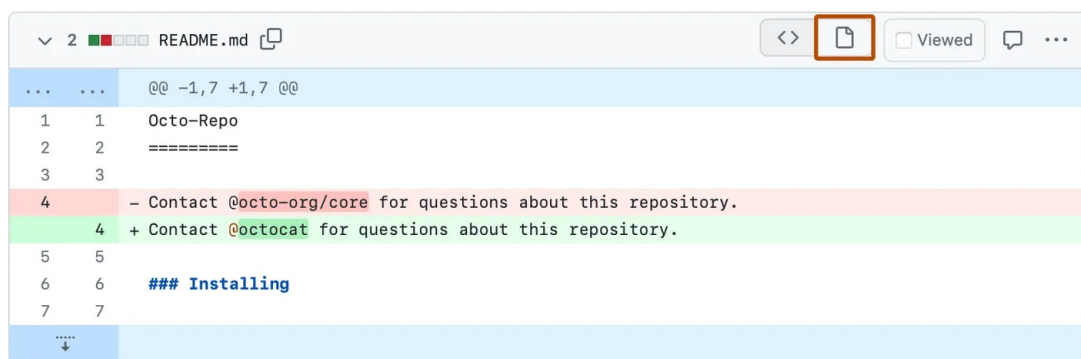
those documents with *source* and *rendered* views.

The source view shows the raw text that has been typed, while the rendered view shows how that text would look once it's rendered on GitHub Enterprise Server. For example, this might be the difference between showing `**bold**` in Markdown, and **bold** in the rendered view.

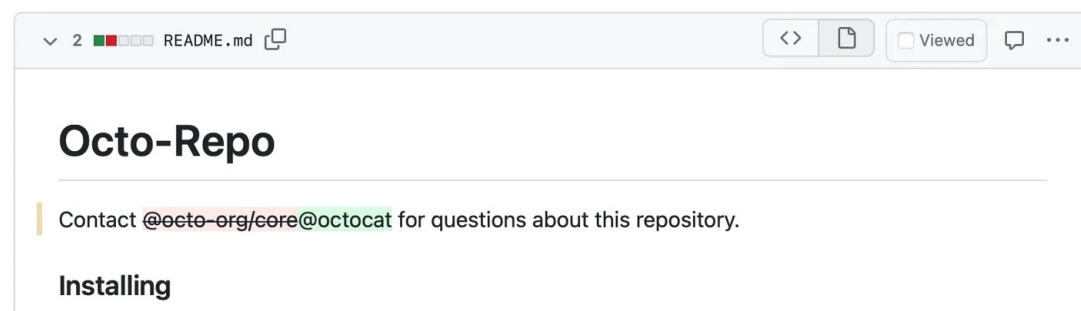
Prose rendering is supported for rendered documents supported by [github/markup](https://github.com/github/markup):

- Markdown
- AsciiDoc
- Textile
- ReStructuredText
- Rdoc
- Org
- Creole
- MediaWiki
- Pod

To see the changes made to the document as part of a commit, click .



This "rich diff" highlights the code that has been added and removed.



Disabling Markdown rendering [↗](#)

When viewing a Markdown file, you can click `<>` at the top of the file to disable Markdown rendering and view the file's source instead.

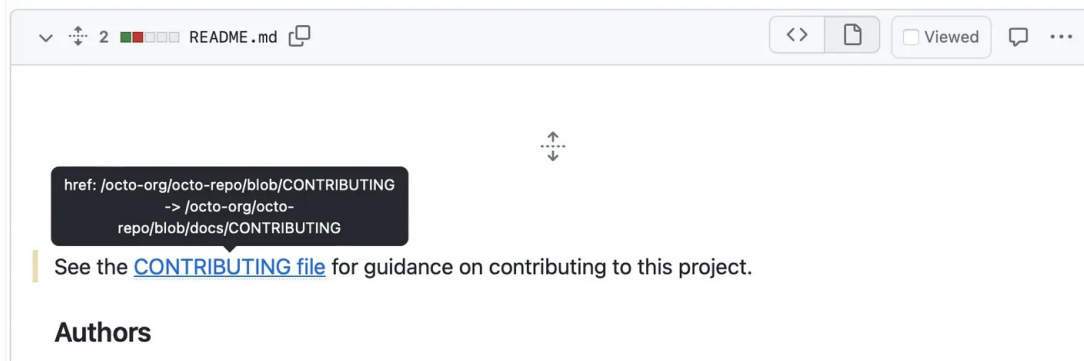


Disabling Markdown rendering enables you to use source view features, such as line linking, which is not possible when viewing rendered Markdown files.

Visualizing attribute changes [↗](#)

We provide a tooltip describing changes to attributes that, unlike words, would not otherwise be visible in the rendered document. For example, if a link URL changes from

one website to another, we'd show a tooltip like this:



Commenting on changes [↗](#)

[Commit comments](#) can only be added to files within the *source* view, on a line-by-line basis.

Linking to headers [↗](#)

As with [other rendered prose documents](#), hovering over a header in your document creates a link icon. You can link readers of your rendered prose diff to specific sections.

Viewing complex diffs [↗](#)

Some pull requests involve a large number of changes with large, complex documents. When the changes take too long to analyze, GitHub Enterprise Server can't always produce a rendered view of the changes. If this happens, you'll see an error message when you click the rendered button.

You can still use the source view to analyze and comment on changes.

Viewing HTML elements [↗](#)

We don't directly support rendered views of commits to HTML documents. Some formats, such as Markdown, let you embed arbitrary HTML in a document. When these documents are shown on GitHub Enterprise Server, some of that embedded HTML can be shown in a preview, while some (like an embedded YouTube video) cannot.

In general, rendered views of changes to a document containing embedded HTML will show changes to the elements that are supported in GitHub Enterprise Server's view of the document. Changes to documents containing embedded HTML should always be reviewed in both the rendered and source views for completeness.

Mapping GeoJSON/TopoJSON files on GitHub [↗](#)

GitHub Enterprise Server supports rendering GeoJSON and TopoJSON map files within GitHub Enterprise Server repositories. Commit the file as you would normally using a `.geojson` or `.topojson` extension. Files with a `.json` extension are also supported, but only if `type` is set to `FeatureCollection`, `GeometryCollection`, or `topology`. Then, navigate to the path of the GeoJSON/TopoJSON file on GitHub Enterprise Server.

Geometry types [↗](#)

Maps on GitHub Enterprise Server use [Leaflet.js](#) and support all the geometry types outlined in [the geoJSON spec](#) (Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection). TopoJSON files should be type "Topology" and

adhere to the [TopoJSON spec](#).

Styling features

You can customize the way features are displayed, such as specifying a particular color or adding a descriptive icon, by passing additional metadata within the GeoJSON object's properties. The options are:

- `marker-size` - `small`, `medium`, or `large`
- `marker-color` - valid RGB hex color
- `marker-symbol` - an icon ID from [the Maki project](#) or a single alphanumeric character (a-z or 0-9).
- `stroke` - color of a polygon edge or line (RGB)
- `stroke-opacity` - opacity of a polygon edge or line (0.0 - 1.0)
- `stroke-width` - width of a polygon edge or line
- `fill` - the color of the interior of a polygon (GRB)
- `fill-opacity` - the opacity of the interior of a polygon (0.0-1.0)

See [version 1.1.0 of the open simplestyle spec](#) for more information.

Embedding your map elsewhere

Want to make your GeoJSON map available someplace other than GitHub Enterprise Server? Simply modify this template, and place it in any HTML page that supports JavaScript (for example, [GitHub Pages](#)):

```
<script
src="https://embed.github.com/view/geojson/<username>/<repo>/<ref>/<path_to_file>">
</script>
```

For example, if your map's URL is [github.com/benbalter/dc-wifi-social/blob/master/bars.geojson](#), your embed code would be:

```
<script src="https://embed.github.com/view/geojson/benbalter/dc-wifi-social/master/bars.geojson"></script>
```

By default, the embedded map 420px x 620px, but you can customize the output by passing height and width variables as parameters at the end, such as `?height=300&width=500`.

Note: `ref` can be a branch or the hash to an individual commit (like `2391ae`).

Mapping in Markdown

You can embed GeoJSON and TopoJSON directly in Markdown. For more information, see "[Creating diagrams](#)."

To display interactive maps, a site administrator must configure the feature for your GitHub Enterprise Server instance. For more information, see "[Configuring applications](#)."

Clustering

If your map contains a large number of markers (roughly over 750), GitHub will automatically cluster nearby markers at higher zoom levels. Simply click the cluster or zoom in to see individual markers.

Something's up with the underlying map [↗](#)

The underlying map data (street names, roads, etc.) are driven by [OpenStreetMap](#), a collaborative project to create a free editable map of the world. If you notice something's not quite right, since it's open source, simply [sign up](#) and submit a fix.

Troubleshooting GeoJSON/TopoJSON files [↗](#)

If you're having trouble rendering GeoJSON files, ensure you have a valid GeoJSON file by running it through a [GeoJSON linter](#). If your points aren't appearing where you'd expect (for example, in the middle of the ocean), it's likely that the data is in a projection which is currently unsupported. Currently, GitHub Enterprise Server only supports the `urn:ogc:def:crs:OGC:1.3:CRS84` projection.

Additionally, if your `.geojson` file is especially large (over 10 MB), it is not possible to render within the browser. If that's the case, you'll generally see a message that says we can't show files that large.

It may still be possible to render the data by converting the `.geojson` file to [TopoJSON](#), a compression format that, in some cases, can reduce filesize by up to 80%. Of course, you can always break the file into smaller chunks (such as by state or by year), and store the data as multiple files within the repository.

Further reading about GeoJSON/TopoJSON [↗](#)

- [Leaflet.js documentation](#)
- [MapBox marker-styling documentation](#)
- [TopoJSON Wiki](#)

Working with Jupyter Notebook files on GitHub [↗](#)

When you add Jupyter Notebook or IPython Notebook files with a `.ipynb` extension on your GitHub Enterprise Server instance, they will render as static HTML files in your repository.

The interactive features of the notebook, such as custom JavaScript plots, will not work in your repository on your GitHub Enterprise Server instance. For an example, see [Linking and Interactions.ipynb](#).

To view your Jupyter notebook with JavaScript content rendered or to share your notebook files with others you can use [nbviewer](#). For an example, see [Linking and Interactions.ipynb](#) rendered on nbviewer.

To view a fully interactive version of your Jupyter Notebook, you can set up a notebook server locally. For more information, see [Jupyter's official documentation](#).

Troubleshooting Jupyter Notebook files [↗](#)

If you're having trouble rendering Jupyter Notebook files in static HTML, you can convert the file locally on the command line by using the `nbconvert` command:

```
jupyter nbconvert --to html NOTEBOOK-NAME.ipynb
```

Further reading about Jupyter Notebook [↗](#)

- [Jupyter Notebook's GitHub repository](#)

- [Gallery of Jupyter Notebooks](#)

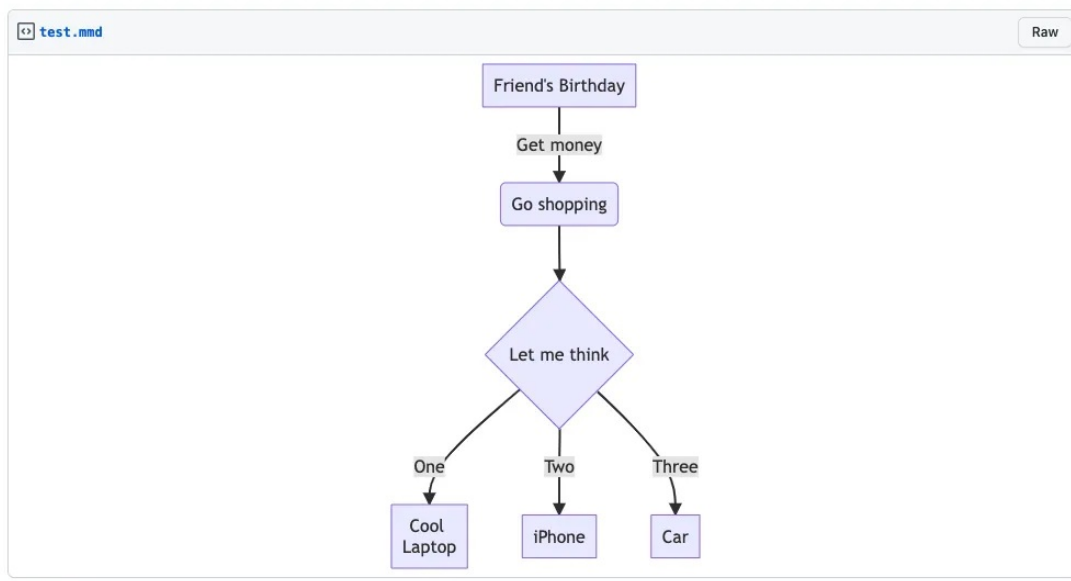
Displaying Mermaid files on GitHub [↗](#)

GitHub Enterprise Server supports rendering Mermaid files within repositories. Commit the file as you would normally using a `.mermaid` or `.mmd` extension. Then, navigate to the path of the Mermaid file on GitHub.

For example, if you add a `.mmd` file with the following content to your repository:

```
graph TD
  A[Friend's Birthday] -->|Get money| B(Go shopping)
  B --> C{Let me think}
  C -->|One| D["Cool <br> Laptop"]
  C -->|Two| E[iPhone]
  C -->|Three| F[fa:fa-car Car]
```

When you view the file in the repository, it is rendered as a flow chart.



Troubleshooting Mermaid files [↗](#)

If your chart does not render at all, verify that it contains valid Mermaid Markdown syntax by checking your chart with the [Mermaid live editor](#).

If the chart displays, but does not appear as you'd expect, you can create a new [GitHub Community discussion](#), and add the `Mermaid` label.

Known issues [↗](#)

- Sequence diagram charts frequently render with additional padding below the chart, with more padding added as the chart size increases. This is a known issue with the Mermaid library.
- Actor nodes with popover menus do not work as expected within sequence diagram charts. This is due to a discrepancy in how JavaScript events are added to a chart when the Mermaid library's API is used to render a chart.
- Not all charts are a11y compliant. This may affect users who rely on a screen reader.

Mermaid in Markdown [↗](#)

You can embed Mermaid syntax directly in Markdown. For more information, see "[Creating diagrams](#)."

Further reading about Mermaid

- [Mermaid.js documentation](#)
- [Mermaid.js live editor](#)

Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)