

Unsere Story editieren

Als gemeinsamen Speicherort für unsere Geschichte, habe ich uns ein GitHub Repository eingerichtet. GitHub ist ein Tool, dass Programmierer benutzen um mit mehreren Leuten an einem Programm zu arbeiten und dabei nicht durcheinander zu kommen.

Unser GitHub Repo befindet sich hier: https://github.com/stauchen/ithiriell_story

Um die Geschichte bearbeiten zu können braucht ihr als erstes einen GitHub Account. Den könnt ihr euch einfach hier anlegen: <https://github.com/>

Wenn ihr das getan habt, dann **sagt mir bescheid**, damit ich euch Schreibrechte auf unserem Repo geben kann. Nachdem das erledigt ist, richtet den Editor ein, mit dem ihr die Geschichte bearbeiten und eure Änderungen ausprobieren könnt.

Gitpod

GitPod ist ein Browserprogramm was unsere Story von Github herunterladen, bearbeiten und Änderungen wieder veröffentlichen kann. 50 Stunden pro Monat ist die Nutzung davon kostenlos und damit sollten wir, denke ich, auskommen. Um Gitpod zu verwenden müssen wir es mit Github verbinden. Ruft zunächst mal folgende Seite auf:

https://gitpod.io/#https://github.com/stauchen/ithiriell_story

Und drückt den freundlichen, blauen "Login with Github & Launch Workspace" Knopf. Das bringt Euch zur Loginseite von Github selbst. Meldet euch dort mit eurem Account an ("Sign in") und klickt auf "Authotize gitpod-io" um dem Editor Zugriff auf eure Githubdateien zu geben. Dann fragt Github euch noch, ob ihr noch weitere Rechte verleihen wollt. Akzeptiert auch das. Und dann startet auch schon der Editor. Das kann ein paar Minütchen dauern.

Auf der linken Seite des Editors sehr ihr die Dateien in unserem Repo. Die wichtigsten davon liegen im "src" Verzeichnis. Das sind die .tweet Dateien, in der unsere Geschichte beschrieben wird. Schaut zum Beispiel unter "src", "examples", "examples.tweet" an. Wenn ihr doppelt auf die Datei klickt, dann öffnet sie sich im Hauptfenster. Ansonsten ist das Verzeichnis "images" noch wichtig. Da drin liegen die Bilder, die wir in der Story verwenden können.

Unten ist die Kommandozeile. Die müsst ihr erst mal nicht weiter beachten. Was ihr dort seht ist eine Internetadresse ("<https://8000-d9594c1a-922f-462f-af9b-473db932d1d5.ws-eu01.gitpod.io>") in meinem Fall. Die ist bei jedem ein bisschen anders. Wenn ihr auf der Tastatur "STRG", oder beim Mac den Teppichklopfer, gedrückt haltet und da drauf klickt öffnet sich die Seite in einem neuen Reiter. Das ist die Geschichte, wie sie gerade bei euch in der Bearbeitung ist. Okay. Ändern wir den Text in der ersten Beschreibung. Geht zurück in den Editor in die "examples.tweet". In Zeile 8 steht

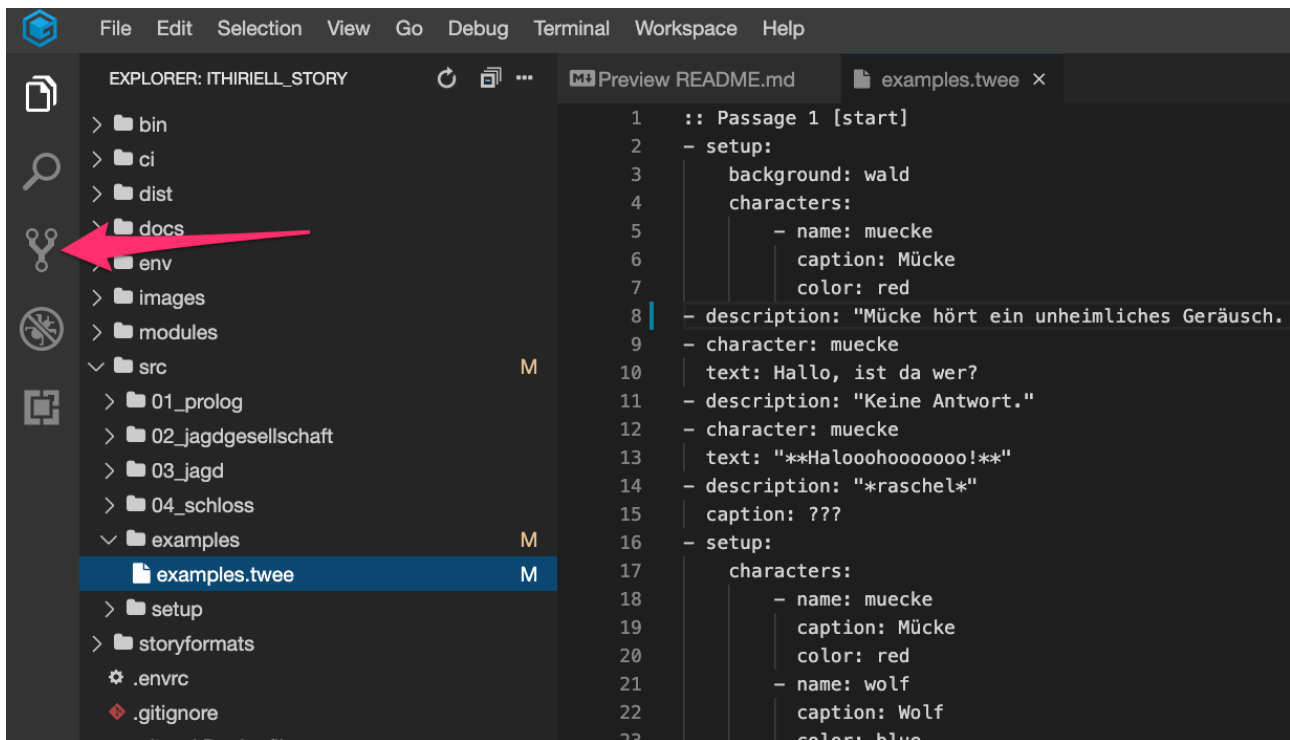
```
- description: "Mücke hört ein unheimliches Geräusch. Da ist noch jemand  
  anderes im Wald ... "
```

Oder etwas ähnliches. Ändert den Satz ab, zum Beispiel so:

```
- description: "Mücke hört ein lautes Geräusch. Da ist noch jemand  
  anderes im Wald ... "
```

Drückt dann STRG+s (beziehungsweise Teppichklopfer+s) um die Datei zu speichern. Geht wieder in den Reiter mit der Geschichte (oder klickt noch mal auf dem Link in der Kommandozeile, falls ihr die Seite wieder geschlossen habt) und ladet die Seite neu (STRG+R, Teppichklopfer+R). Voilà der Text hat sich geändert!

Zurück im Texteditor seht ihr in der Dateiliste angezeigt, welche Dateien ihr verändert habt. Die sind mit einem "M" markiert. Lasst uns für den Moment eure Änderung rückgängig machen. Später erkläre ich euch, wie ihr Änderungen in eurer eigenen Kopie der Geschichte machen könnt und wie ihr die Änderungen wieder in die Hauptversion der Geschichte zurück fließen lasst. Um die Änderung rückgängig zu machen, klickt ganz links vom Editor auf das Symbol, was eine Verzweigung zeigt:



Dort findet ihr unter "Changes" die "examples.twee" aufgelistet. Wenn ihr auf den Eintrag klickt, geht im Hauptfenster ein neuer Reiter auf, der euch die Unterschiede zwischen eurer Version und der Hauptversion zeigt.

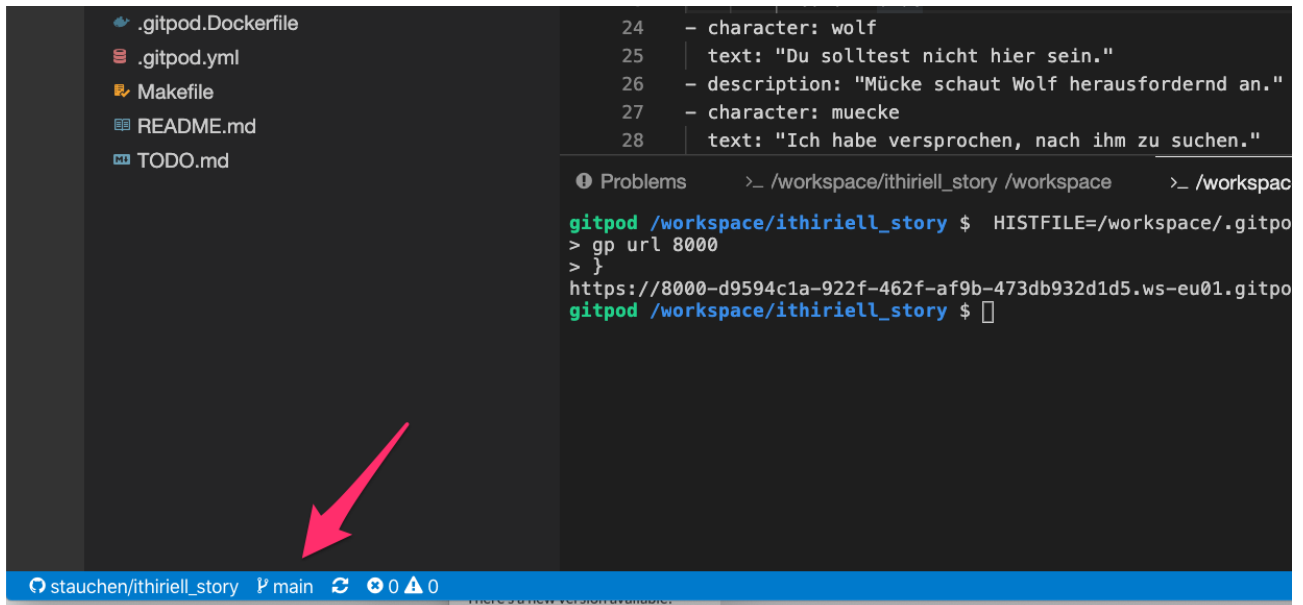
Wenn ihr mit der Maus links unter Changes auf "examples.twee" stehen bleibt tauchen rechts ein paar Schaltflächen auf.

Die erste (der Pfeil, der gekrümmt gegen den Uhrzeigersinn geht) nimmt eure Änderung zurück. Das gleiche kann man auch mit allen Änderungen machen, falls man mehr als eine Datei geändert hat, in dem man die Schaltfläche rechts neben "CHANGES" anwählt. Klickt auf den Pfeil und bestätigt den Dialog, dann ist die Datei wieder so wie vorher. Man kann also nichts kaputt machen, wenn man herum ändert. Und man kommt immer wieder auf einen Zustand, der irgendwann mal funktioniert hat. Im Hauptfenster könnt ihr auch den Reiter mit dem Vergleich beider Versionen schließen in dem ihr in der Kopfzeile auf das kleine "x" drückt. Wählt links über dem Verzweigungssymbol das Symbol, dass zwei geknickte Din-A4 Seiten zeigt und ihr gelangt zurück in die Dateiübersicht.

Branching

Damit wir uns mit unseren Änderungen nicht gegenseitig auf den Füßen herum latschen, kann man eigene Versionen der Geschichte anlegen, in denen man zunächst mal alleine seine Änderungen macht. Diese unterschiedlichen Versionen heißen bei git "Branches" (Verzweigungen)

In der blauen Fußzeile des Editors findet ihr wieder ein Verzweigungssymbol, neben dem "main" steht. Das zeigt euch an, welche Version der Geschichte ihr gerade anschaut. "Main" ist die Hauptversion der Geschichte.



```
24 - character: wolf
25   text: "Du solltest nicht hier sein."
26 - description: "Mücke schaut Wolf herausfordernd an."
27 - character: muecke
28   text: "Ich habe versprochen, nach ihm zu suchen."

gitpod /workspace/ithiriell_story $ HISTFILE=/workspace/.gitpod
> gp url 8000
> }
https://8000-d9594c1a-922f-462f-af9b-473db932d1d5.ws-eu01.gitpod
gitpod /workspace/ithiriell_story $
```

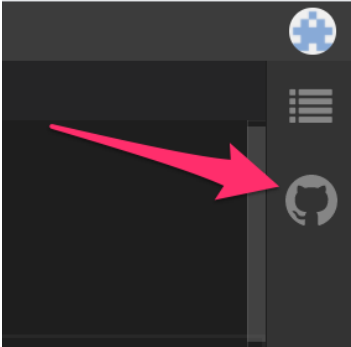
stauchen/ithiriell_story main

Wenn ihr da drauf klickt, bekommt ihr eine Liste von anderen Versionen der Geschichte angezeigt. Genauso wie die Option "Create new branch....". Klickt auf "Create new branch...." und gebt ihm einen schönen Namen. Ich nenne meinen "cisco-test" und drücke Enter. Am besten benennt man die Branches danach, was man in ihnen ändern möchte. Beispielsweise "Kap2/GesprächMitRost". Branchnamen dürfen keine Leerzeichen enthalten.

Nun gut. Ändern wir noch einmal den ersten Satz in der examples.twee, wie oben und navigieren wieder links mit dem Verzweigungssymbol zu den "Changes". Diesmal wollen wir die Changes unserer eigenen Version der Geschichte hinzufügen. Statt auf den Pfeil, der die Änderung zurück nimmt klickt diesmal auf das "+", entweder neben der Datei oder neben CHANGES um alle Änderungen zu akzeptieren. Damit kann man genau aussuchen, welche Änderung man überhaupt übernehmen und veröffentlichen möchte. Gebt in dem Kopffeld eine Beschreibung eurer Änderung ein. Hier kann man hinterlassen, warum man etwas geändert hat, falls man sich in ein paar Tagen fragt, was man sich bei dieser Änderung eigentlich gedacht hat. Ich schreibe "Besseres Adjektiv gefunden". Dann drückt man STRG+Enter (Teppichklopfer+Enter) oder drückt auf das Häkchen über dem Textfeld um die Änderungen einzufügen. Gitbenutzer sprechen hierbei von einem "Commit". So ein "Commit" besteht aus mehreren zusammenhängenden Dateiänderungen. Das macht man alles so kleinschrittig, damit man nachvollziehen kann, wer wann was geändert hat.

Push

Um die Änderung zu publizieren müssen wir sie nach Github übermitteln. Diesen Vorgang nennt man "Push". Umgekehrt, wenn man Änderungen von Github holt, weil man, beispielsweise zu zweit an einem Branch arbeitet, nennt man das einen "Pull". Ganz rechts im Fenster sehr ihr ein Symbol was die Silhouette einer Katze zeigt. (Das ist Octocat, das Logo von Github)



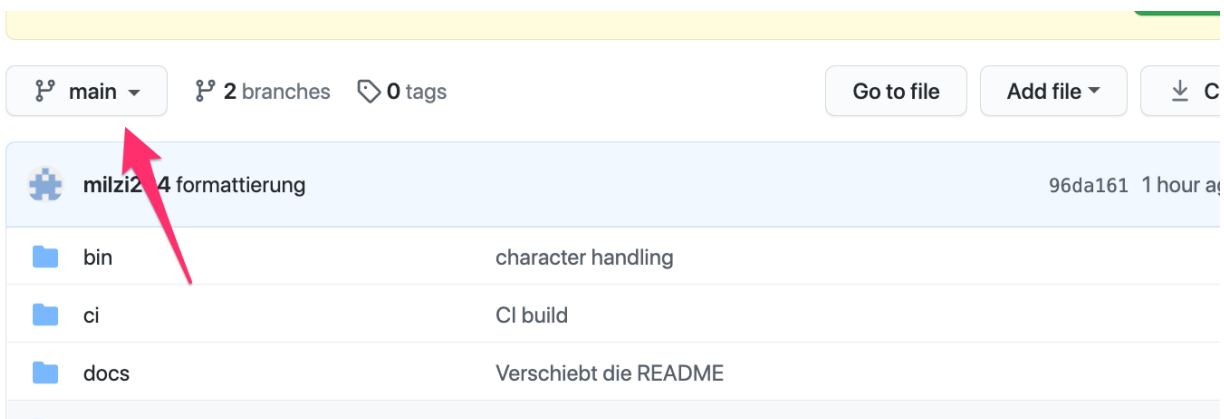
Wenn ihr da drauf klickt, öffnet ihr ein neues Teilfenster in dem ihr mit Github interagieren könnt. "Publish Changes" kopiert euren Branch nach Github. Klickt da drauf und bestätigt den Dialog. Das dauert etwa eine Sekunde in der der Button stehen bleibt. Nach einer Weile ändert sich die Ansicht. Das ignorieren wir für den Moment. Weiter geht es auf Github.

Änderungen zusammenführen

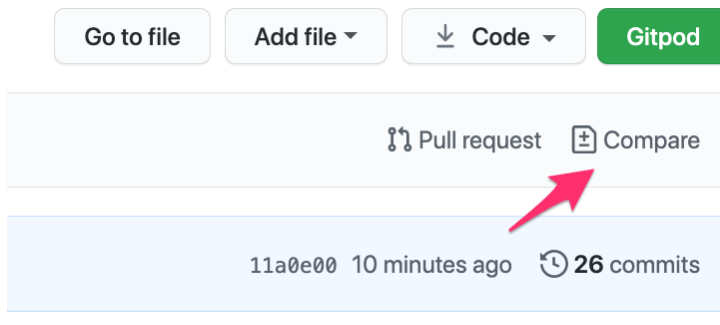
Um die Änderung in unserem Branch mit dem "main" Branch zusammen zu führen erstellen wir einen sog. Pull Request. Geht auf

https://github.com/stauchen/ithiriell_story

Dort seht ihr über der Dateiliste ein Dropdownmenü in dem "main" steht. Wenn ihr da drauf klickt, seht ihr auch euren Branch. Klickt da drauf und wählt ihn aus.



Dann taucht rechts ein "Compare" Button auf. Mit dem gelangt ihr in eine Ansicht, die eure Version und die Hauptversion der Geschichte miteinander vergleicht. Auch da könnt ihr noch mal ganz genau eure Änderungen sehen.



Klickt da drauf und dann in der nächsten Ansicht, nachdem ihr euch ein wenig umgeschaut habt, wählt "**Create pull request**". Hier könnt ihr noch ein bisschen ausführlicher beschreiben, was ihr an der Geschichte geändert habt. Für unsere kleine Änderung ist das nicht nötig. Dann wählt "**Create Pull Request**" um den Request anzulegen. Dann passieren allerlei Dinge, die uns hier erst mal nicht beschäftigen sollen. (Aber schaut euch ruhig die Sachen mal an, die da alle auftauchen! Wenn ihr Fragen über die einzelnen Sachen habt und mehr wissen wollt, dann fragt mich gerne!) Wenn eure Änderung ohne Schwierigkeiten mit der Hauptversion der Geschichte zusammengeführt werden kann, dann erscheint weiter unten ein Knopf "**Merge pull request**" und danach auf "**Confirm merge**". "Merge" meint hier das zusammenführen zweier Branches.

Jetzt könnt ihr noch euren Branch wegräumen mit "**Delete branch**", denn den brauchen wir jetzt nicht mehr, da eure Änderung in der Hauptgeschichte aufgegangen ist. Nach kurzer Zeit (manchmal sofort, manchmal erst nach einer halben Stunde) wird eure Änderung auch in der publizierten Version der Geschichte bei <https://ithiriell.stauchen-stories.com/> auftauchen.

Zu guter letzt müssen wir noch den Editor abschalten. Sonst geht das die ganze Zeit gegen unser 50-Stunden-Kontingent. Öffnet diese Website: <https://gitpod.io/workspaces/>. Die listet eure Arbeitsbereiche auf. Wählt dort "Stop" um den Arbeitsbereich zu schließen. Von hier aus könnt ihr einen Arbeitsbereich auch wieder starten. Das ist nützlich, wenn er Änderungen enthält, die ihr noch nicht committet oder gepusht habt.

Ihr könnt einen neuen Arbeitsbereich erstellen, in dem ihr wieder https://gitpod.io/#https://github.com/stauchen/ithiriell_story aufruft. Das soll es erst mal für heute sein.

Wenn ihr wissen wollt, wie die Syntax für die .twee Dateien ist, dann schaut mal hier:

https://github.com/stauchen/ithiriell_story/blob/main/docs/01_Syntax.md

Die erste Passage, die angesprungen wird, ist die, die ein [start] Tag enthält. Also probiert mal das [start] Tag von der Beispielpassage wegzunehmen und eine eigene Passage einzufügen, mit dem Tag. Oder ändert ein bisschen in der Beispielpassage herum.

Wenn Fragen auftauchen, oder ihr irgendwo stecken bleibt, dann lasst es mich wissen!