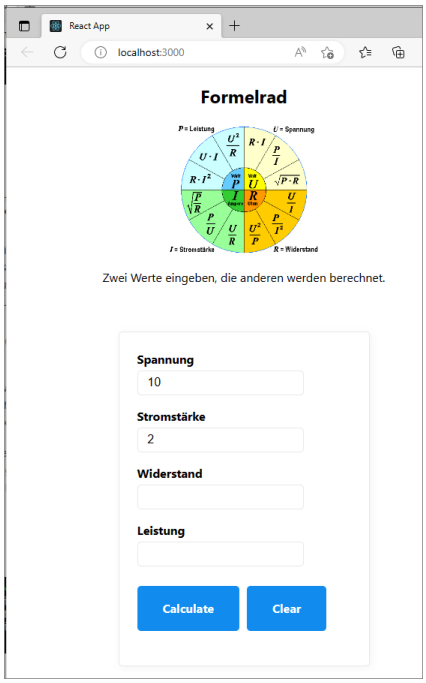
Anwenden von Commit, Merge und …

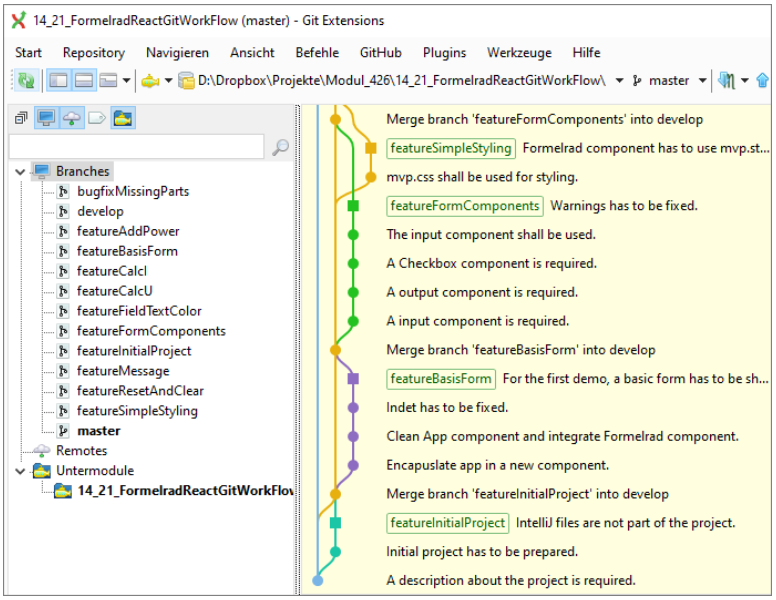
Einleitung:

Voraussetzung für diese Übung sind grundlegende Kenntnisse zu **Git, dessen Befehle** sowie zum Workflow von **git-flow** notwendig.

In dieser Übung geht es nun darum, dass Sie git, commit, merge, branch … intensiv anwenden. Sie werden Merge-Konflikte lösen. Und Sie werden weiter den Process von git-flow umsetzten.

Sie werden die Möglichkeit Softwareteile via Patches zu importieren kennen lernen und anwenden. Und so Schritt um Schritt die React-Applikation Formelrad aufbauen und es wird ein Git-Baum mit mehreren Branches entstehen.





Ziel

- ⇨ Sie wenden git und seine Befehle intensiv an.
- ⇨ Sie wenden den Workflow von Git-flow intensiv an.
- ⇨ Sie lösen Merge-Konflikte.
- ⇨ Sie wenden git Patches an und erkennen dabei, was ein Commit eigentlich speichert.

Inhalte

Aufgabe: Repository vorbereiten

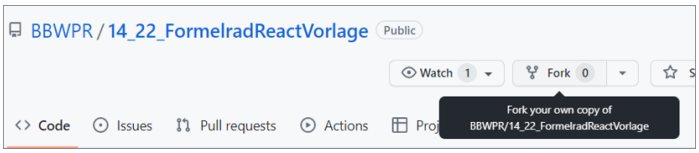
Sie übernehmen das auf github vorbreitetet Projekt.
Dazu wenden Sie **git fork** and, damit Sie Besitzer von Ihrem Repository werden und über Ihr Repository frei verfügen können, ohne das vorbereitet Repository zu verändern.

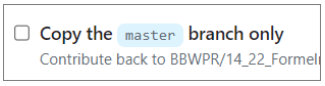
Was der Unterschied von git clone und git fork ist, wird in diesem Video erklärt.

https://www.youtube.com/watch?v=6YQxkxw8nhE

Aufgabe:

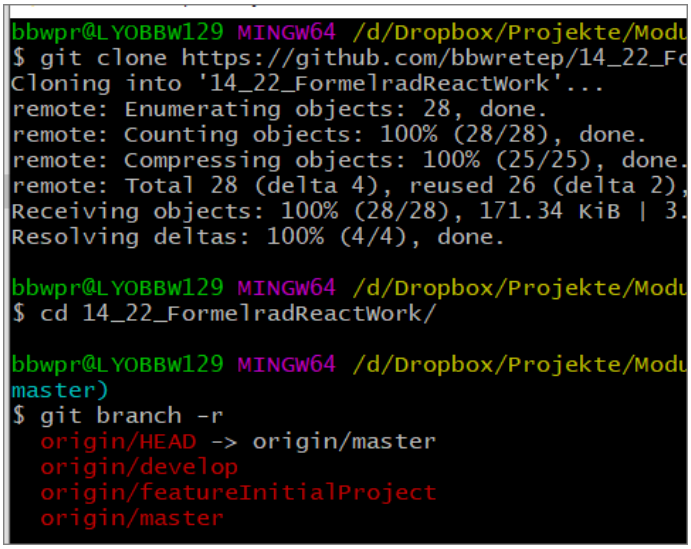
- Machen Sie eine Fork vom Projekt
  Übernehmen Sie dabei alle Branches.



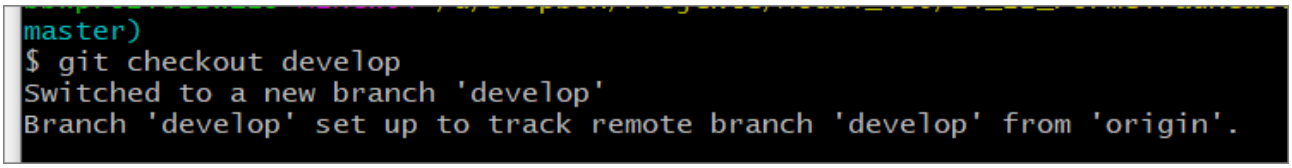


https://github.com/BBWPR/14_22_FormelradReactVorlage

- Clonen Sie Ihr Projekt auf Ihren Rechner



- Wechseln Sie auf den develop branch.



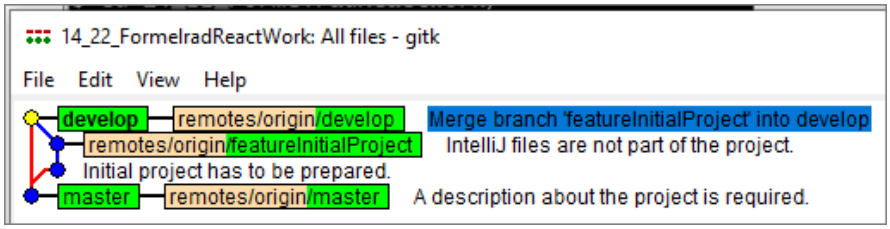
- Starten Sie das Projekt

```
npm install
npm start
```

Input zu den existierenden Branches

Sie können den Git-Baum mit Hilfe von **gitk** ausgeben.

```
//Alle branches lokal pull
git fetch --all
git pull origin master
git pull origin develop
git pull origin featureInitialProject

gitk
```



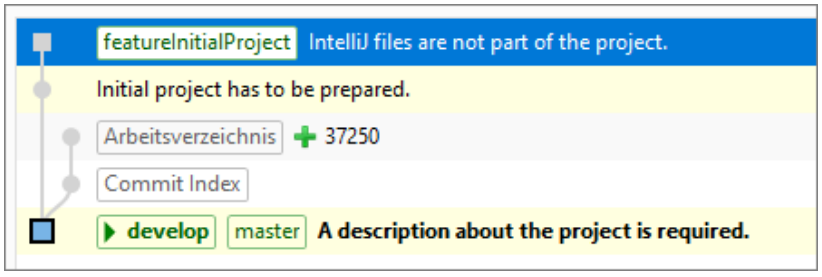
(Tools wie **git extension** zeigen den Baum noch detaillierter)

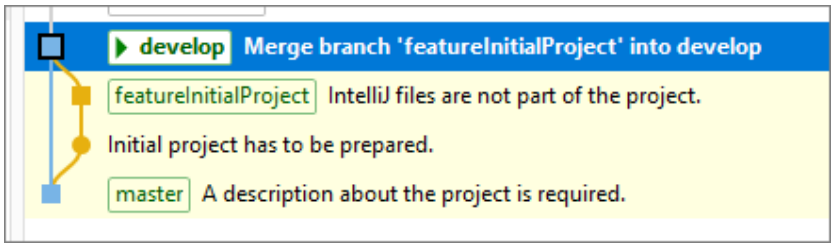
Sie sehen, wie im Projekt bereits ein *master, ein develop* und ein *featureInitialProject* Branch existieren.

Beachten Sie:

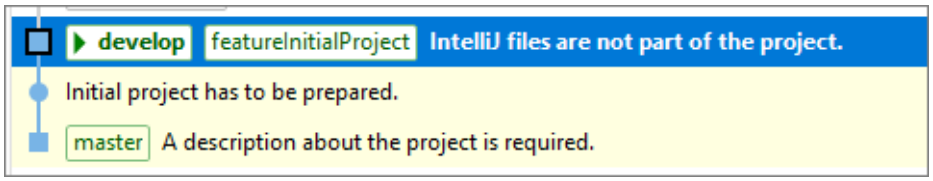
Der Branch featureInitialProject wurde bereits mit *develop merge* ***--no-ff*** gemerged.
Dadurch bleibt der featureInitialProject sichtbar.

Wirkung von --no-ff





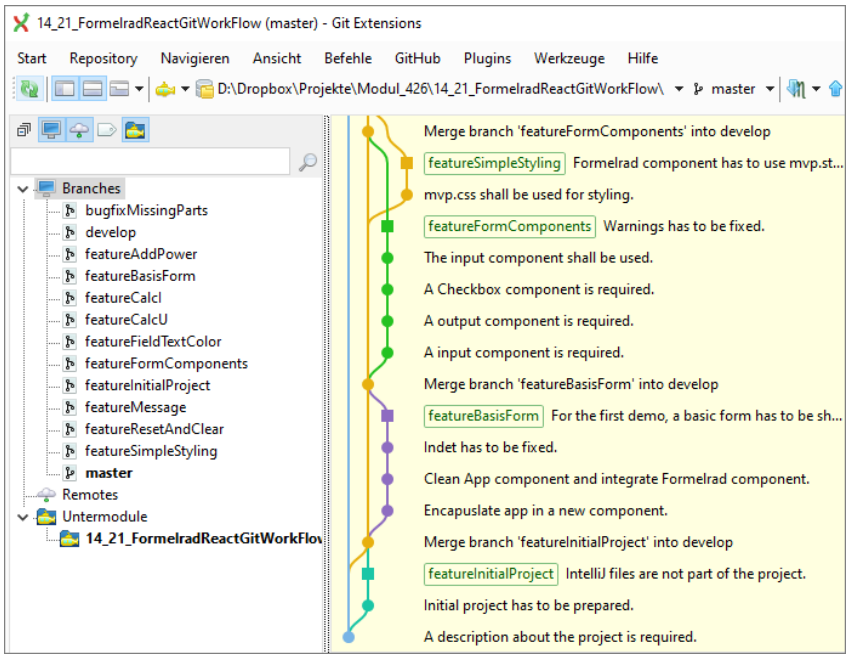
Ohne --no-ff

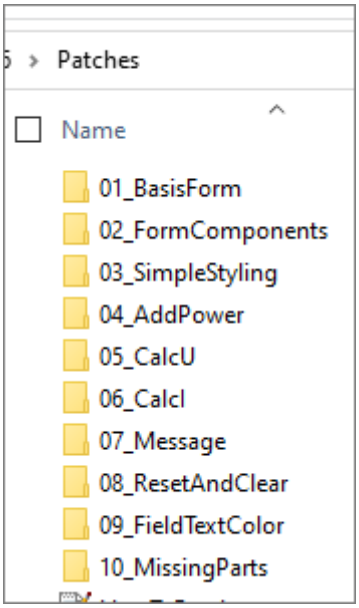


Verwenden Sie in der Übung den merge Befehl immer mit --no-ff.

Aufgabe: Mit Hilfe von Patches den Baum aufbauen

Sie werden nun mit Hilfe git und Patches den Projektbaum aufspannen.



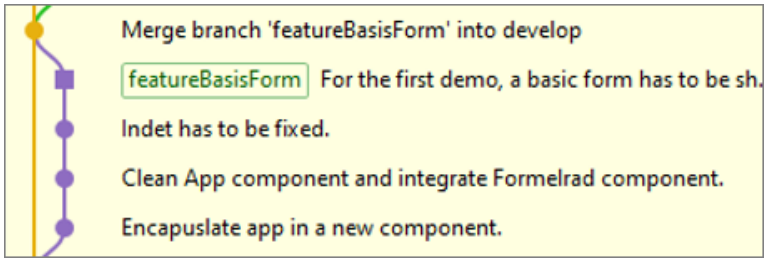


Aufgabe Vorbereitung:

- Bereiten Sie die Patches aus dem ZIP-Patch-Archiv in einem separaten Verzeichnis vor.

Aufgabe erster Feature Branch:

Sie sehen oben, dass mit einem nächsten Feature Branch die Basis für das Formular gelegt wird.



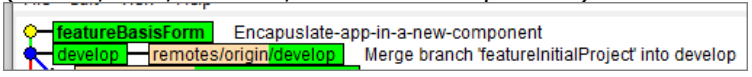
- In den *Develop* Branch wechseln.
- Den neuen Feature-Branch eröffnen z.B. *git checkout -b FeatureBasisForm*
- Den ersten Patch anwenden
  Nutzen Sie den Tab, um File Namen zu vervollständigen

  ```
  git apply --keep ../Patches/01_BasisForm/0001-<TAB>
  ```

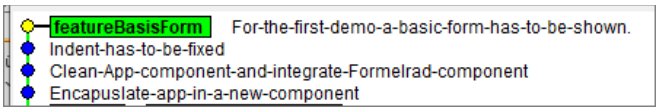
  wird zu

  ```
  git apply --keep ../Patches/01_BasisForm/0001-Encapuslate-app-in-a-new-component.patch
  ```

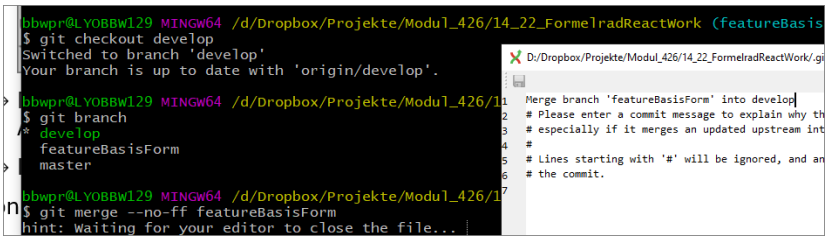
- Der Patch ist angelegt. Nun müssen Sie in *"Commiten"* (status, add, commit, status .. und pushen)



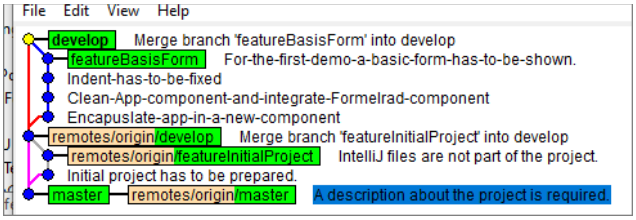
- Wiederholen Sie den Vorgang bis und mit Patch 0004-..
- Resultat:



- Nun müssen Sie den Branch in develop merge.
  Achten Sie auf --no-ff.



- Resultat:

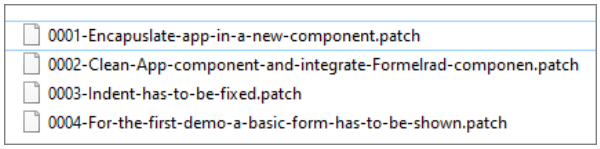


- Kontrollieren Sie, ob das Projekt läuft.

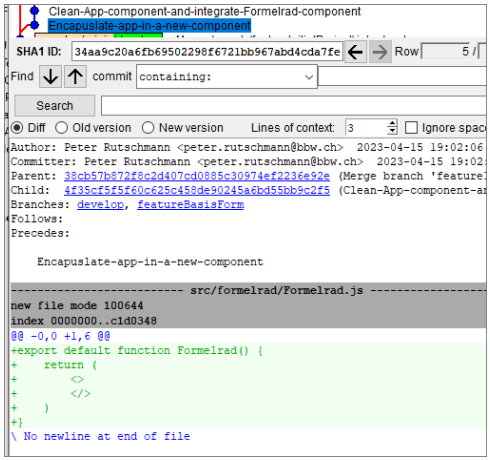
Input:

Schauen Sie sich mal die soeben verwendeten Patch-Dateien mit einem Editor an.

In reiner Textform ist die Veränderung, die der Patch bewirkt, beschrieben.



Vergleichen Sie mit dem, was gitk zu einem Commit ausgibt:



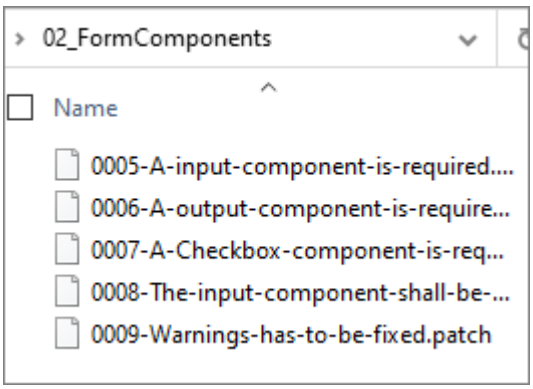
Was stellen Sie fest?

Aufgabe: Weitere Patches anwenden, erster Merge Konflikt beheben.

Sie simulieren nun zwei Entwickler, die gleichzeitig je ein Feature implementieren.

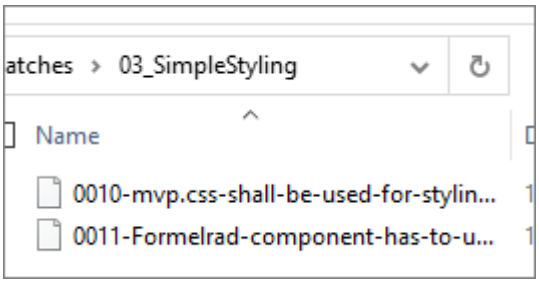
Aufgabe zweiter Feature Branch:

- Machen Sie am Schluss keinen Merge, gehen Sie ansonsten gleich vor.
  Erstellen Sie den zweiten Branch auf der Basis der Patches aus



Aufgabe dritter Feature Branch:

- Wechseln Sie in den develop branch, der neue Branch muss vom develop aus starten.
- Machen Sie am Schluss keinen Merge, gehen Sie ansonsten gleich vor.
  Erstellen Sie den dritten Branch auf der Basis der Patches aus



Aufgabe zweiter Feature Branch mergen

- Wechseln Sie in den develop branch.
  Mergen Sie nun den zweiten Feature Branch in develop.
  --no-ff nicht vergessen.

Aufgabe dritter Feature Branch mergen

- Mergen Sie nun den dritter Feature Branch in develop.
  --no-ff nicht vergessen.

Und es wird einen Merge Konflikt geben. Siehe nächste Seite…

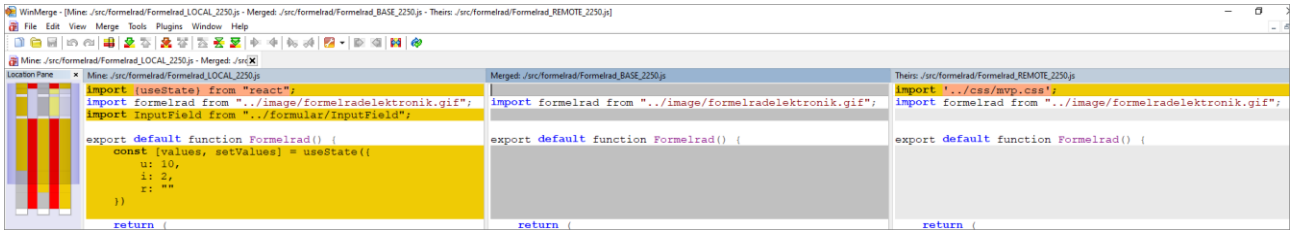
```
bbwpr@LYOBBW129 MINGW64 /d/Dropbox/Projekte/Modul_426/14_22_FormelradReactWork (f
$ git checkout develop
Switched to branch 'develop'
Your branch is ahead of 'origin/develop' by 5 commits.
  (use "git push" to publish your local commits)

bbwpr@LYOBBW129 MINGW64 /d/Dropbox/Projekte/Modul_426/14_22_FormelradReactWork (d
$ git branch
* develop
  featureBasisForm
  featureFormComponents
  featureSimpleStyling
  master

bbwpr@LYOBBW129 MINGW64 /d/Dropbox/Projekte/Modul_426/14_22_FormelradReactWork (d
$ git merge --no-ff featureFormComponents
Merge made by the 'recursive' strategy.
 src/App.js                  |  1 -
 src/formelrad/Formelrad.js  | 20 ++++++++++----------
 src/formular/Checkbox.js    | 15 +++++++++++++++
 src/formular/InputField.js  | 14 ++++++++++++++
 src/formular/OutputField.js | 12 ++++++++++++
 5 files changed, 52 insertions(+), 10 deletions(-)
 create mode 100644 src/formular/Checkbox.js
 create mode 100644 src/formular/InputField.js
 create mode 100644 src/formular/OutputField.js

bbwpr@LYOBBW129 MINGW64 /d/Dropbox/Projekte/Modul_426/14_22_FormelradReactWork (d
$ git merge --no-ff featureSimpleStyling
Auto-merging src/formelrad/Formelrad.js
CONFLICT (content): Merge conflict in src/formelrad/Formelrad.js
Automatic merge failed; fix conflicts and then commit the result.

bbwpr@LYOBBW129 MINGW64 /d/Dropbox/Projekte/Modul_426/14_22_FormelradReactWork (d
```

Aufgabe informieren:

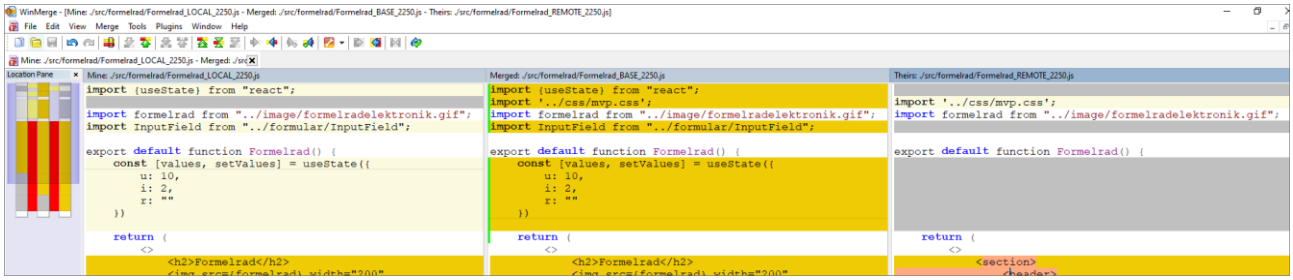
- Lesen Sie in der Datei *14.6_MergenUndKonflickte.pdf* nach, wie Sie einen Merge Konflikt lösen können.

Hier nur ein kurzer Auszug, was ich mache…

Mergetool aufrufen und Unterschiede analysieren.



Erste Teile sind von Hand richtig merged:

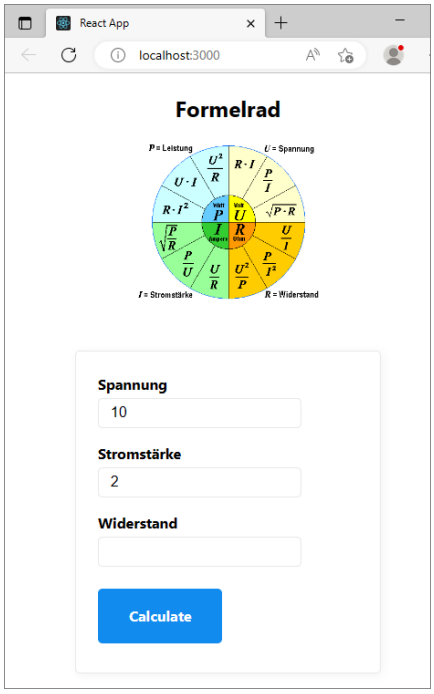


Der untere Teil richtig gemerged:

```
    return (
        <>
            <section>
                <header>

                    <h2>Formelrad</h2>
                    <img src={formelrad} width="200" alt="Formelrad"/>



                </header>


                <form><InputField color={"black"} value={values.u}
label="Spannung" handleChange={e => {setValues(values => ({...values, u:
e.target.value}))}} />

                    <InputField color={"black"} value={values.i}
label="StromstÃ¤rke" handleChange={e => {setValues(values => ({...values, i:
e.target.value}))}} />
                    <InputField color={"black"} value={values.r}
label="Widerstand" handleChange={e => {setValues(values => ({...values, r:
e.target.value}))}} />
                    <button type="submit">Calculate</button>
                </form>
            </section>
```

Wenn Sie das Projekt gestartet haben, können Sie laufend das Ergebnis überprüfen.

Und wenn alles richtig ist...

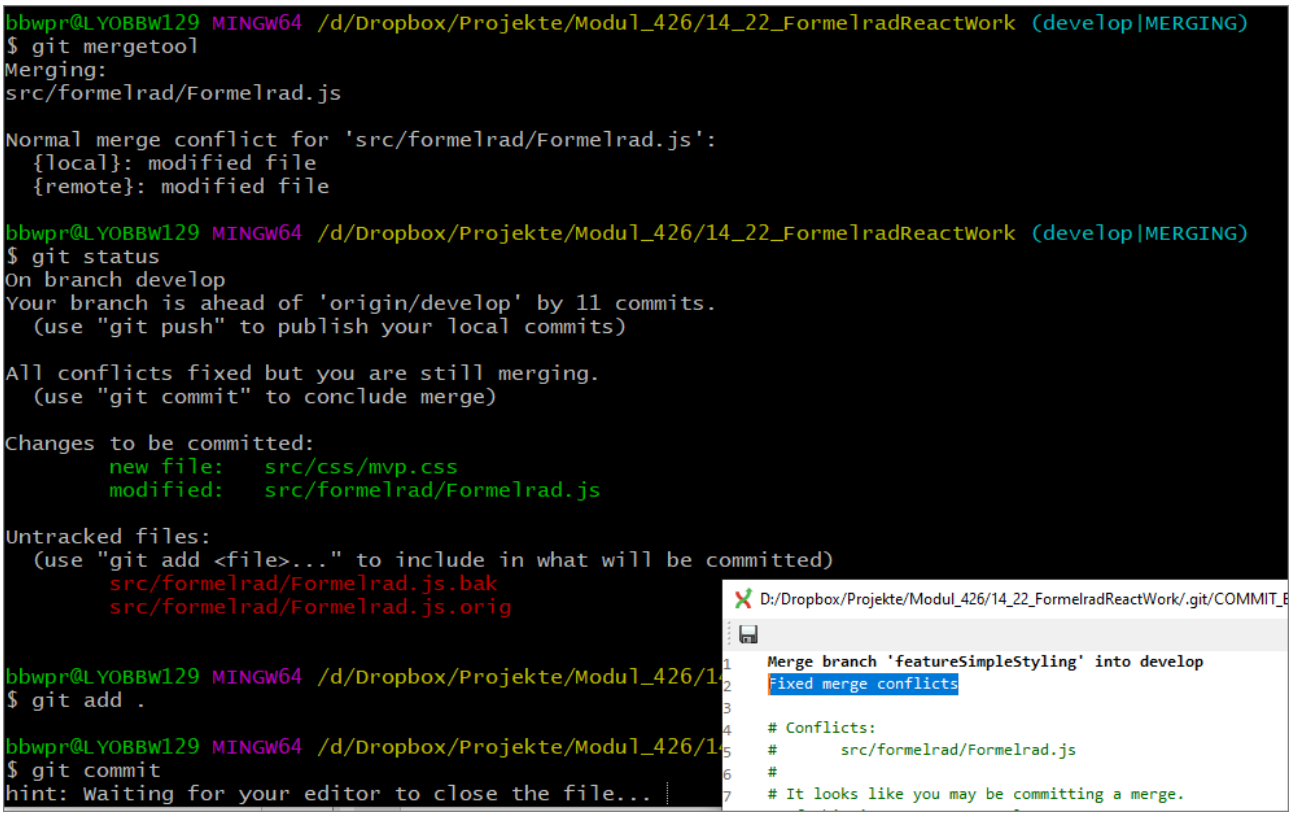


Aufgabe: Lösen Sie den Merge Konflikt

- Unterschiede bereinigen
- status, add
- ACHTUNG commit ohne -a und -m
  Da dank --no-ff bereits eine Commit-Message vorbereitet ist.

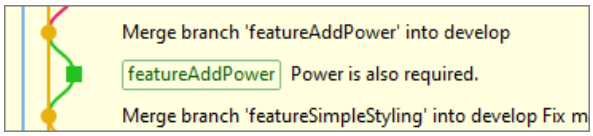
```
bbwpr@LYOBBW129 MINGW64 /d/Dropbox/Projekte/Modul_426/14_2
$ git commit
hint: Waiting for your editor to close the file...
```

Ganzer Ablauf inklusive commit:



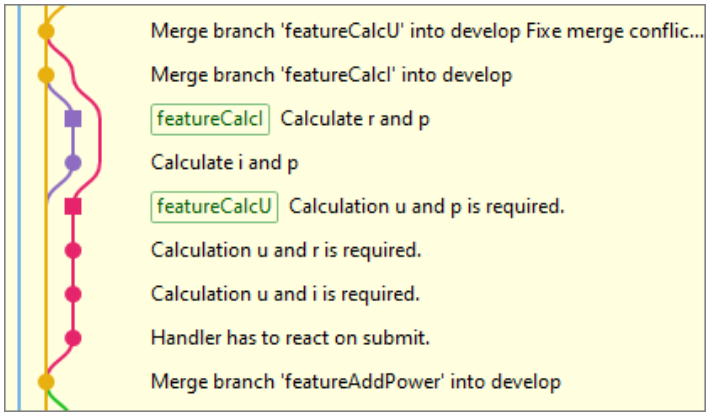
Aufgabe: Patches AddPower anwenden .

Wenden Sie die Patches für das feature AddPower an.



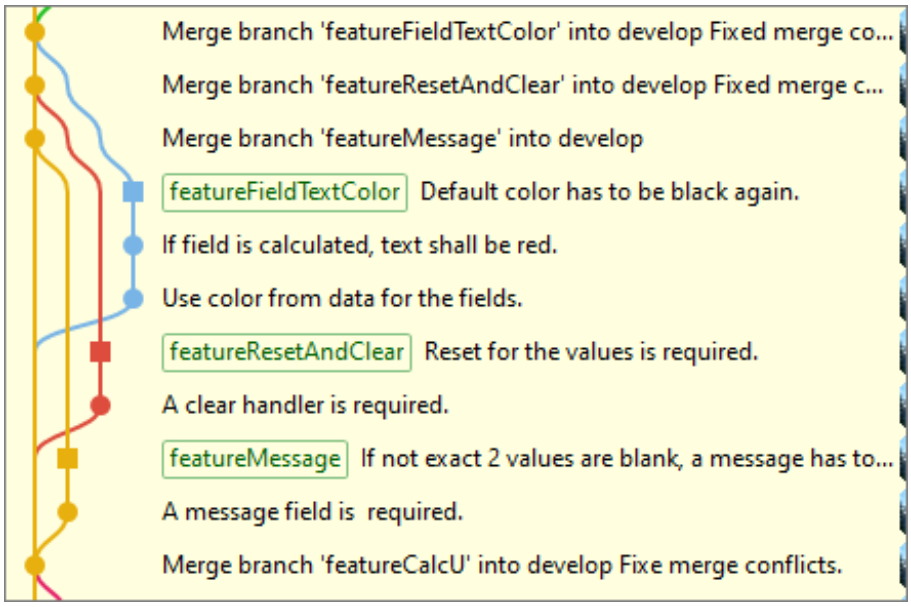
Aufgabe: Patches CalcU und CalcI anwenden .

Sie simulieren wiederum zwei Entwickler.
Wenden Sie die Patches für das feature CalcU und CalcI an ohne zu Mergen.
Mergen Sie am Schluss und lösen Sie den Merge Konflikt.



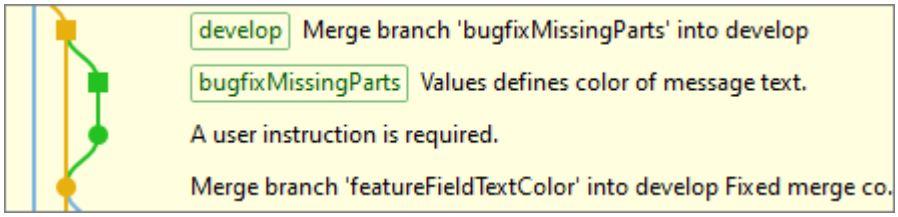
Zusatz-Aufgabe: Patches Message, ResetAndClear und FieldTextColor anwenden.

Sie simulieren drei Entwickler.
Wenden Sie die Patches auf drei neue Branches an ohne sofort Mergen.
Mergen Sie am Schluss und lösen Sie den Merge Konflikt.



Zusatz-Aufgabe: Bugfix Patch anwenden.

Es wird noch ein Bug gefixt, also kein Feature.



Zusatz-Aufgabe: Merge in master.

Damit ist die Version 1.0.0 fertig.

Mergen Sie den develop Branch in den Master Branch.

Tagen Sie den Master Branch mit der Version 1.0.0