# ReDI Digital Heroes

## Web Developer Fundamentals

Grundlagen für angehende Web-Entwickler*innen

Stand: 2021-04-07

# Inhalt

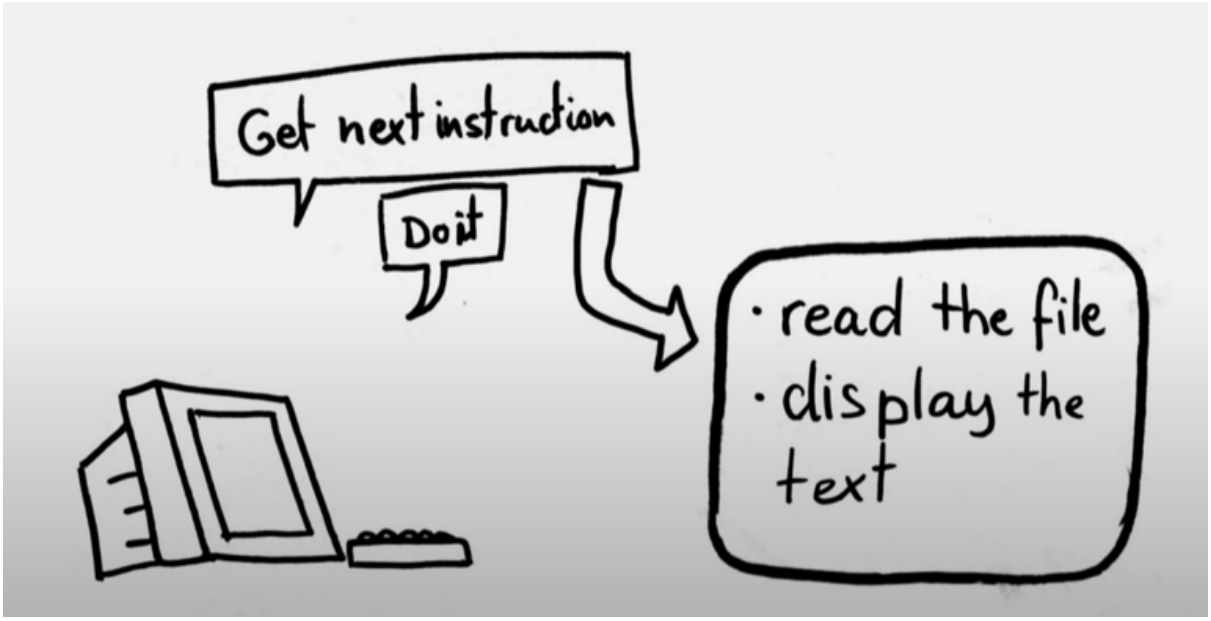
# Basiswissen, Werkzeuge und Teamarbeit

## Entwickler-Basiswissen

Bevor man Web Entwickler werden kann, ist es notwendig erstmal ein paar Grundsätzlichkeiten von Computern zu verstehen.

### Was ist ein Computer?

Im Wort Computer steht das englische verb *to compute* (rechnen). Demnach ist ein Computer eine universelle Rechenmaschine. Er kann jede gewünschte Berechnung für dich anstellen. Ein Computer wartet immer auf die nächste Anweisung und führt diese aus, wartet auf eine weitere Anweisung und führt sie aus usw.




Der Computer ist *Hardware* (z. B. ein Smartphone) und führt also genau das aus was du ihm befiehlst. Diese Befehle bzw. Anweisungen erfolgen dabei durch das Schreiben und Bedienen von *Software* (z. B. eine App).

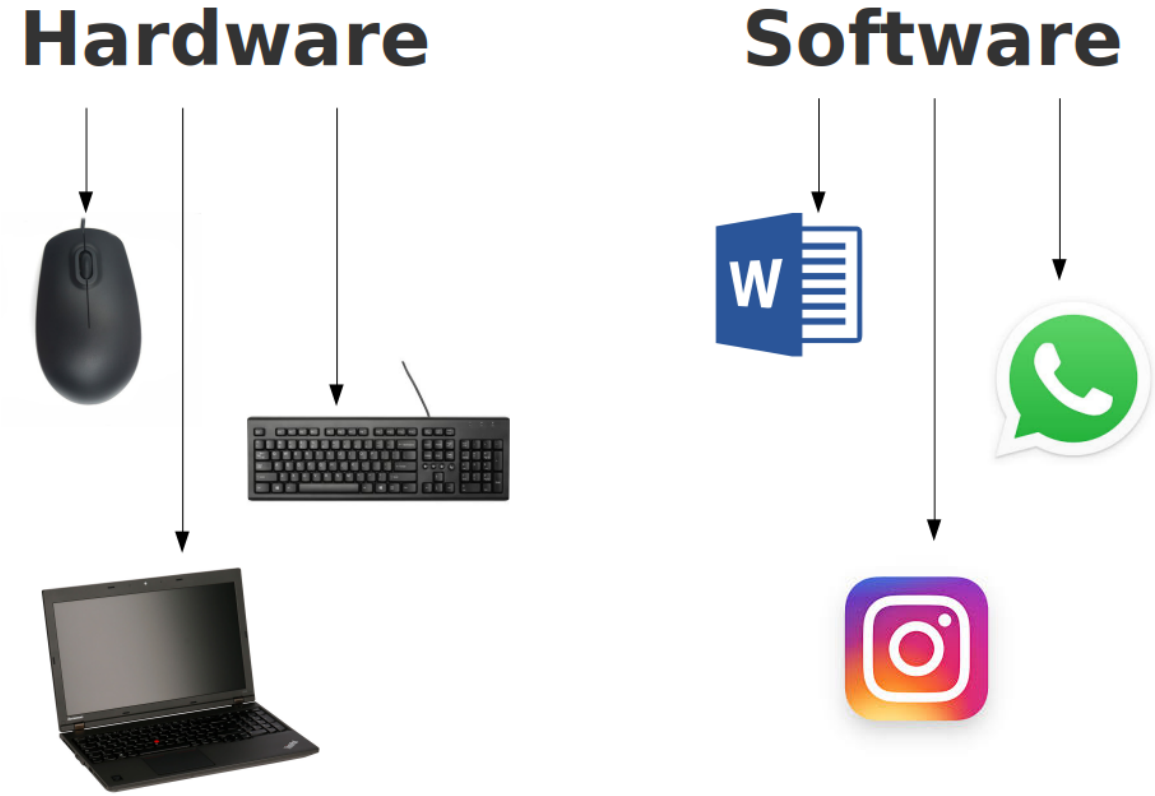
### Hardware und Software

Kurz gesagt sind **Hardware** in der IT die **Dinge, die man anfassen kann**. **Software** dagegen ist **nicht physisch greifbar**. Softwareprogramme (z. B. Apps) sind **Sammlungen von Werten und Anweisungen**.

Das kann man mit einem Rezept vergleichen: Werte (=Zutaten) werden durch Anweisungen in geschriebener Sprache mit der Hardware (=Herd, Kochtopf...) verarbeitet.

Bei der Hardware gibt es neben den Computern selber auch sogenannte **Peripheriegeräte** wie Mäuse, USB-Sticks oder Drucker. Sie **werden an den Computer angeschlossen**.

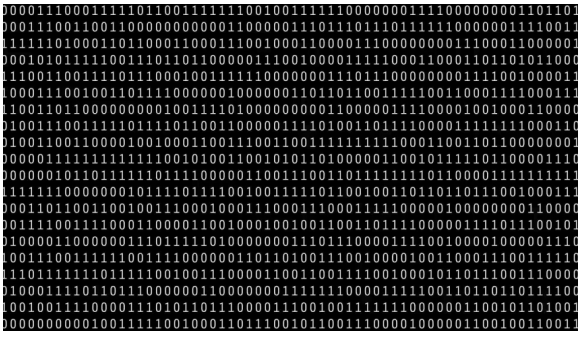
Ein Computer (PC, Laptop, Smartphone, Tablet, Raspberry Pi, Amazon Echo...) ist also Hardware. Peripheriegeräte sind zwar Hardware, aber keine Computer.




## Quellcode und Binärcode

Software kann in zwei Formen in Erscheinung treten: Als menschenlesbarer Quellcode oder als maschinenlesbarer Binärcode.

**Computer verstehen im Grunde nur 0 und 1**. Eine Anweisung in dieser Form für den Computer zu erstellen wäre **viel zu kompliziert für uns Menschen**.



Deswegen schreiben wir diese **Anweisungen in einer für Menschen erlernbaren Form** auf, dem sogenannten **Quellcode**. Dieser Quellcode wird **in einer von vielen Computersprachen** verfasst. Die Computersprachen, die wir in diesem Kurs lernen werden heißen, z. B. *Markdown*, *HTML* und *CSS*.


Es gibt zwei Arten von Computersprachen: *Compilersprachen* und *Interpretersprachen*. Programme, die in Compilersprachen geschrieben wurden, werden immer erst vom Quellcode in Binärcode übersetzt bevor man sie starten kann. So als würdest du einen Zeitungsartikel aus einer anderen Sprache ins Deutsche übersetzen und dann jemanden zum lesen geben.
Bei Interpretersprachen geschieht diese (immer notwendige) Übersetzung automatisch bei der Ausführung. Stell es dir wir einen Simultanübersetzer (Übersetzen während gesprochen wird) vor.

### Was ist ein Server?

Eine simple Frage mit der ihr auch Menschen herausfordern könnt, die sich vermeintlich gut mit Computern auskennen und doch oft falsch beantwortet wird.
Die meisten denken bei Servern immer automatisch an große Computer in Rechenzentren. Das ist aber falsch. **Tatsächlich sind Server Software-Programme (eine App)**. Es können **beliebig viele Server**(-programme) **auf einem Rechner** laufen, so wie du auch mehrere Apps auf deinem Telefon installieren kannst. Auch dein Rechner zu Hause (und sogar ein Smartphone) kann Server-Programme anbieten.
Umgangssprachlich hat sich das Wort Server aber für beides durchgesetzt, da erfahrene Menschen diesen Unterschied genau kennen und immer verstehen wovon gerade geredet wird.

Es handelt sich dabei um zwei Seiten einer Medaille. Ein Server (Diener) macht keinen Sinn ohne einen zugehörigen Client (der Bediente). In unserem Kurs verwenden wir ein Webserver(-programm), welches wir mit dem Client, einem Webbrowser ansteuern.

## Handwerkszeug für Entwickler

In jedem Beruf werden Werkzeuge benötigt, so braucht auch der/die Entwickler*in, ähnlich einem Handwerker bestimmte Werkzeuge mit denen er/sie umgehen können sollte. Wir erlernen dabei die für unseren Kurs entscheidenden.

### Handwerkszeug: Die Tastatur

Die Umgang mit der Tastatur ist entscheidend für einen Entwickler. Auf die Maus solltest du verzichten, solange du Quellcode schreibst.

Die für uns wichtigen Tastaturbefehle habe ich hier zusammengefasst:
https://github.com/liberavia/redi-heroes-keyboard-challenge/blob/main/Einfuehrung1_Tastatur.md
In Kurz: https://bit.ly/2ZGMJ4N

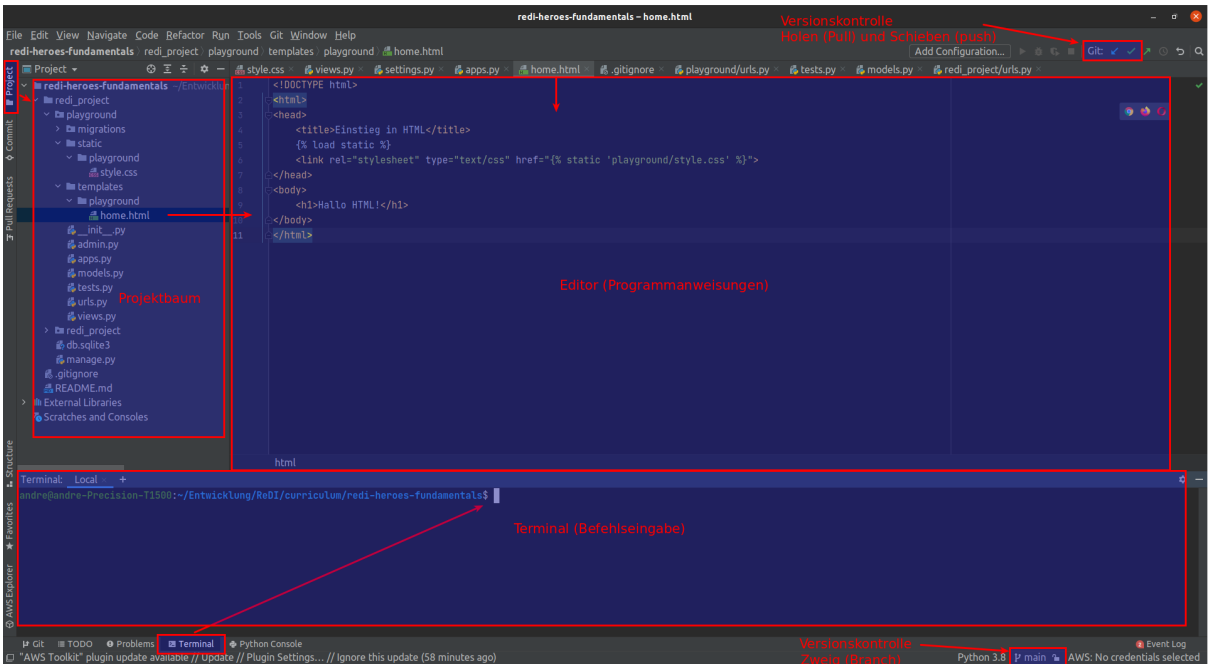
Je öfter du das übst, desto schneller wirst du!

## Handwerkszeug: Die Entwicklungsumgebung (IDE)

Eine Entwicklungsumgebung ist eine Sammlung verschiedener Entwickler-Programme unter einem Dach. Sie hilft bei der Fehlererkennung, markiert das Geschriebene farbig, sorgt für eine Anbindung an ein Versionskontrollsystem, startet Server und vieles mehr. Für Entwicklungsumgebung wird auch gerne das Kürzel IDE verwendet. Das ist die Abkürzung für *Integrated Development Environment*.

### Unsere IDE: PyCharm

Die von uns in diesem Kurs genutzte IDE ist PyCharm. PyCharm ist von der Firma JetBrains und wurde speziell für die Programmiersprache Python entwickelt. Es ist Software die nur im Binärcode vorliegt. Sie kann also nicht näher studiert werden.
PyCharm wird von vielen professionellen Entwicklern verwendet und hat unglaublich viele Funktionen, die selbst von professionellen Entwickler auch nach Jahren der Nutzung nicht alle verwendet werden. Hab also keine Panik. Für uns sind nur einige Elemente wichtig.




### Der Projektbaum

Der Projektbaum befindet sich auf der linken Seite. Eventuell muss er mit einem Klick auf “Projekt” auf der linken Seite.

### Die Editor für Programmieranweisungen

Der größte Teil der IDE ist dem Editor gewidmet. Hier schreiben wir unsere Programmieranweisungen hinein.

### Das Terminal

Mit dem Terminal können wir unserem System befehle geben. Der wichtigste für uns wird der zum Start des Webservers sein:

```
python manage.py runserver
```

Versionskontrolle

Unten rechts sehen wir den Entwicklungszweig in dem wir uns gerade befinden. Mit den Aktionstasten oben rechts können wir uns entweder den aktuellen Stand aus der Versionsverwaltung holen oder unsere Änderungen dorthin übertragen. Was eine Versionskontrolle ist, lernst du im nächsten Abschnitt.

# Zusammenarbeit ohne Streit mit git

*git* ist das meistverwendete Versionskontrollsystem (VCS) der Welt und wurde von *Linus Torvalds* (siehe Bild) entwickelt, der gleichzeitig auch der Erfinder des meistverwendeten Betriebssystemkerns Linux ist.

Mit Versionskontrollsystemen können viele Menschen gleichzeitig an einem Projekt arbeiten. Ohne solche Systeme wäre es einfach nicht möglich, große Projekte mit Menschen rund um den Globus gemeinsam zu stemmen.
Ältere, erfahrene Entwickler*innen können dir sicherlich die eine oder andere Geschichte davon erzählen, welche schlimmen Streits es in Entwicklungsabteilungen gegeben hat, in welchen man keine Versionskontrollsysteme verwendet hat. Es kam dadurch nämlich häufiger vor, dass ein/e Entwickler*in der/dem anderen die Arbeit von Tagen zerstört haben und beide sich dann gemeinsam hinsetzen müssen um das Chaos wieder in Ordnung zu bringen.
In solche Situationen sollten dank git zukünftige Entwickler*innen wie du nicht mehr landen müssen.

## git macht jeden Fehler rückgängig!

git nimmt dir die Angst vor Fehlern. Du kannst nichts kaputt machen und selbst, wenn du es willst, lässt sich der vorherige Stand immer wieder herstellen. Fantastisch, oder? Stell dir das mal im richtigen Leben vor. Du könntest zu jedem beliebigen Punkt in der Zeit zurück und könntest einfach neustarten in einem anderen Zweig des Lebens. In der Programmierung ist dies möglich! 😊

## Entwicklerportale

Theoretisch benötigte man diese Portale nicht wirklich. Sie erleichtern das weltweite Arbeiten allerdings erheblich. Das bekannteste Portal ist GitHub, welches vor wenigen Jahren von der Firma Microsoft gekauft wurde. Es gibt natürlich weitere Portale. Besonders zu erwähnen ist hierbei GitLab, weil es sich um freie Software handelt, die jeder auch auf seinem eigenen Rechner oder Rechenzentrum installieren kannst.
Wir verwenden in unserem Kurs GitHub. Du brauchst dazu keinen eigenen Account. Du bekommst einen von uns.
Wenn du deinen Quelltext, den du dir im Laufe des Kurses erarbeitet hast nach dem Kurs für dich behalten möchtest, helfen wir dir gerne.

## Wie nutze ich mein GitHub-Konto?

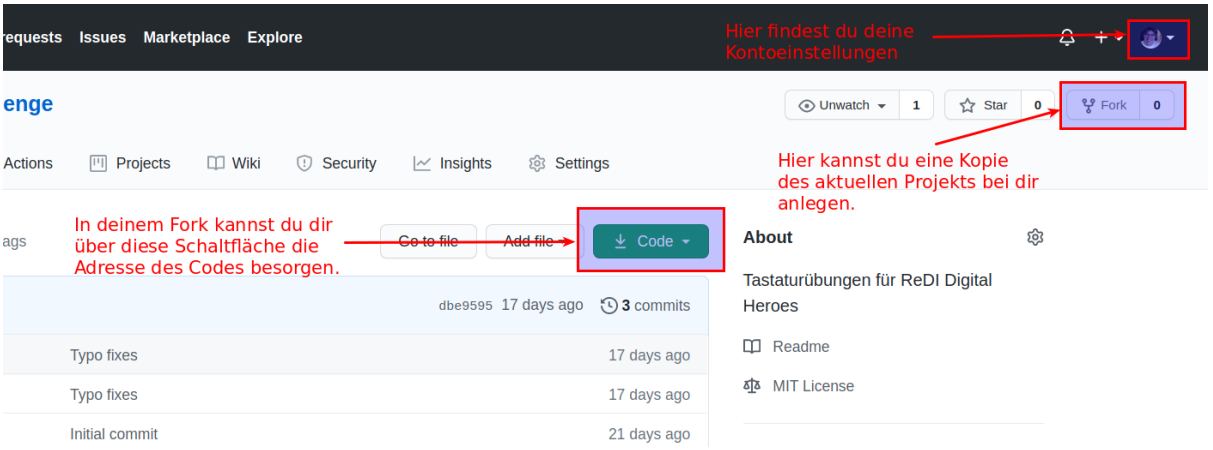
Während des Kurses erhältst du Zugangsdaten von uns, die du während des gesamten Kurses nutzen kannst. Mit diesen Zugangsdaten kannst du dich hier einloggen: https://github.com/login

## Wie arbeite ich mit git? (git-flow)

Mit der git Versionsverwaltung kannst du beliebig viele **Projekte in** Lagerstätten (engl. **Repositories**) verwalten. git merkt sich dabei jede Änderung an einem solchen Repository, so dass du alle Schritte nachvollziehen und zurückspielen kannst. Nichts kann kaputt gehen.

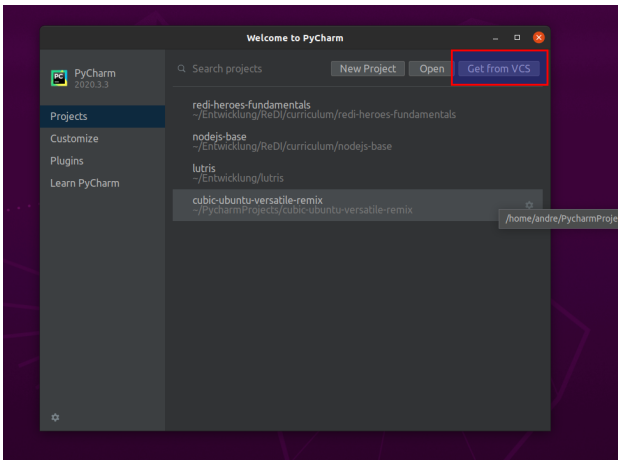
In unserem Kurs arbeitest du mit bereits erstellten vorhandenen Repositories, die du weiter verbesserst. So, wie du mit unseren Beispiel-Repositories arbeitest, kannst du später auch an Millionen Projekten mitwirken. Die Vorgehensweise ist die gleiche.

Schauen dir mal das Repository an mit dem wir zuerst arbeiten: https://github.com/liberavia/redi-heroes-keyboard-challenge

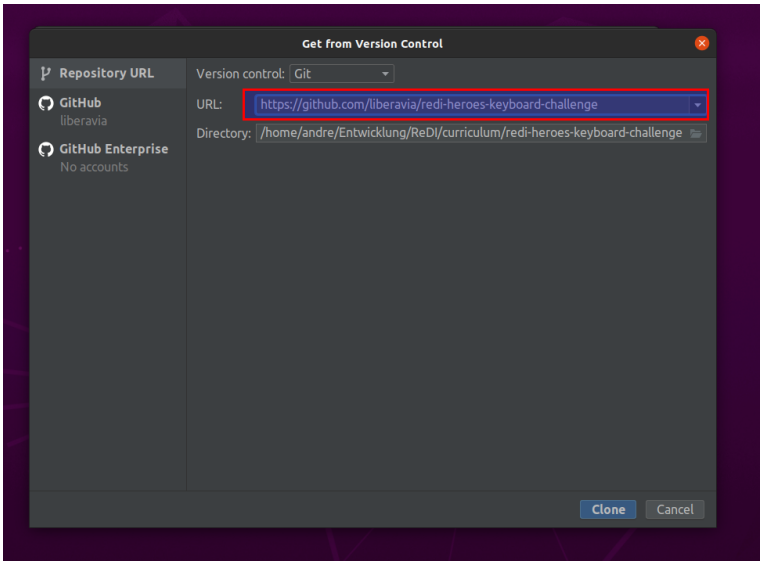



Zuerst erstellst du von dem Repository, bei dem du mitmachen willst, **eine Kopie, die in deinem Konto angelegt wird**. Diese Kopie nennt man einen **Fork** (engl. für Gabel(-ung)).

Um das Projekt dann zu verwenden benutzen wir dazu unsere IDE. Im Startbildschirm gibt es rechts oben einen Knopf “Get From VCS” (engl. Hole es dir vom Versionskontrollsystem):

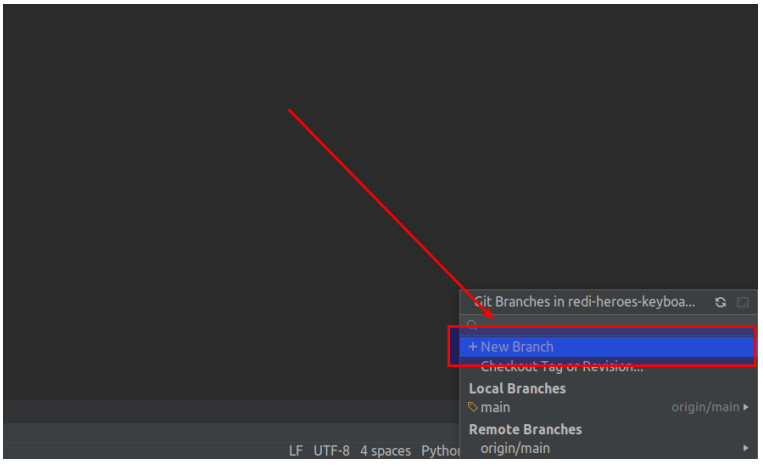


Wenn du hier die Adresse von deinem Fork einträgst, wird **eine Kopie** davon **auf deinem Computer** gespeichert. Diesen Prozess nennt man **Checkout**.

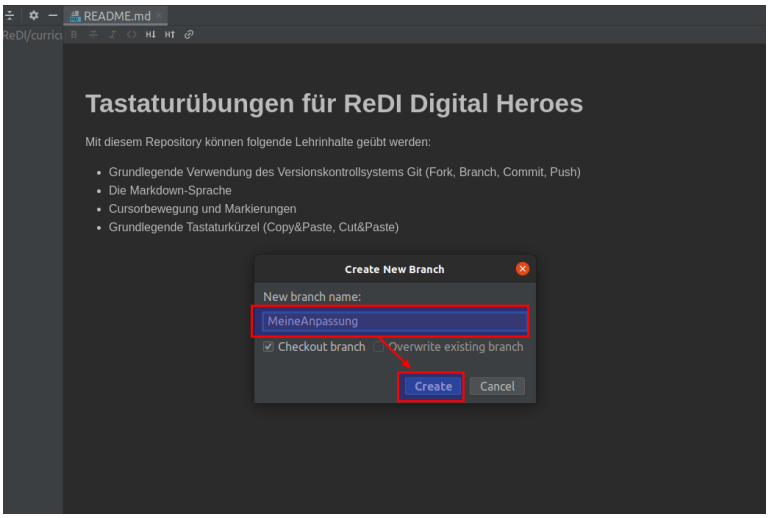



Wenn du nun etwas an dem Projekt ändern willst, erstellst du immer zuerst einen neuen Zweig (engl. **Branch**). Dazu klickst du in der IDE **unten rechts** auf die **Zweigauswahl**:


Klicke dann auf “+ New Branch” um einen neuen Branch zu erstellen:



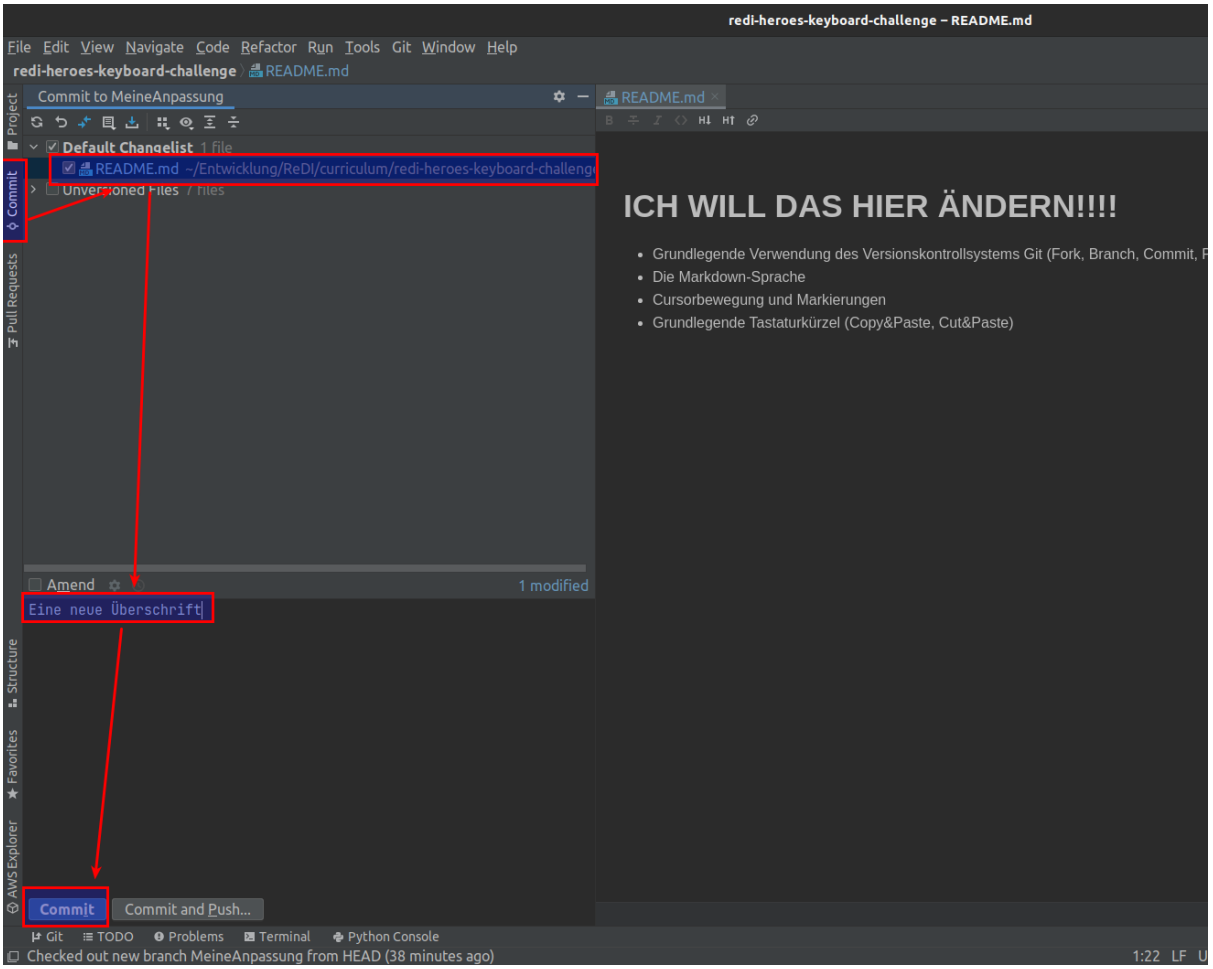
Benenne danach deinen Branch. Du kannst den **Name**n **frei** wählen, darfst **aber keine Leerzeichen und Umlaute** verwenden. Der Name sollte so wie eine gute Überschrift gewählt werden, damit andere und du schnell erkennen können, was hier geändert wird:




Nach dem Klick auf “Create”, kannst du rechts unten in der Zweigauswahl erkennen, dass dein Branch nun aktiv ist:

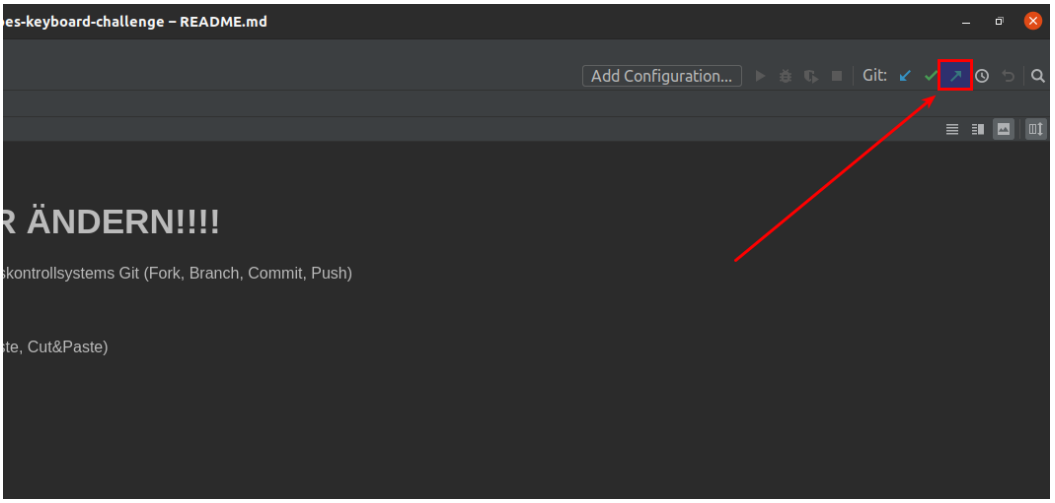

Du kannst über Zweigauswahl auch immer wieder zu einem anderen Zweig hin- und herspringen. Deine Änderungen beeinflussen demnach nichts am Hauptzweig (genannt: main). In deinem Branch kannst du deine Änderungen nun durchführen und du wirst garantiert nichts kaputt machen, selbst wenn du den größten blödsinn machen würdest, wie wahllos Dateien löschen. Mit ein paar Handgriffen ist der alte Stand wieder hergestellt.

Wenn du deine Änderungen dann im Versionskontrollsystem **absichern** willst, erstellst du einen **Commit** (engl. für Vereinbarung). Ein Commit braucht auch immer einen Kommentar, damit du auch Jahre später noch erkennen kannst, was du geändert hast:

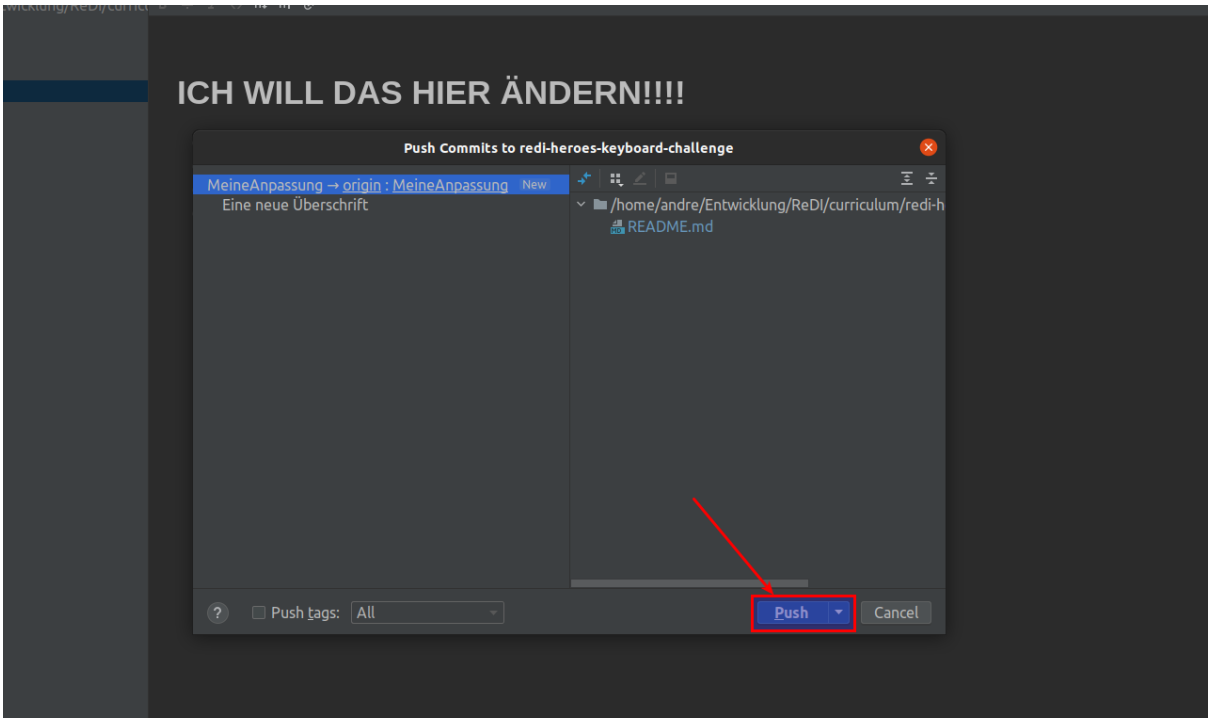



Dies kannst du beliebig oft tun, sobald du etwas geändert hast. Die Änderungen sollten für jeden Commit nicht zu umfangreich sein. Sie sollte den Umfang haben, dass du die Änderung mit einem kurzen Satz im Kommentar beschreiben kannst.

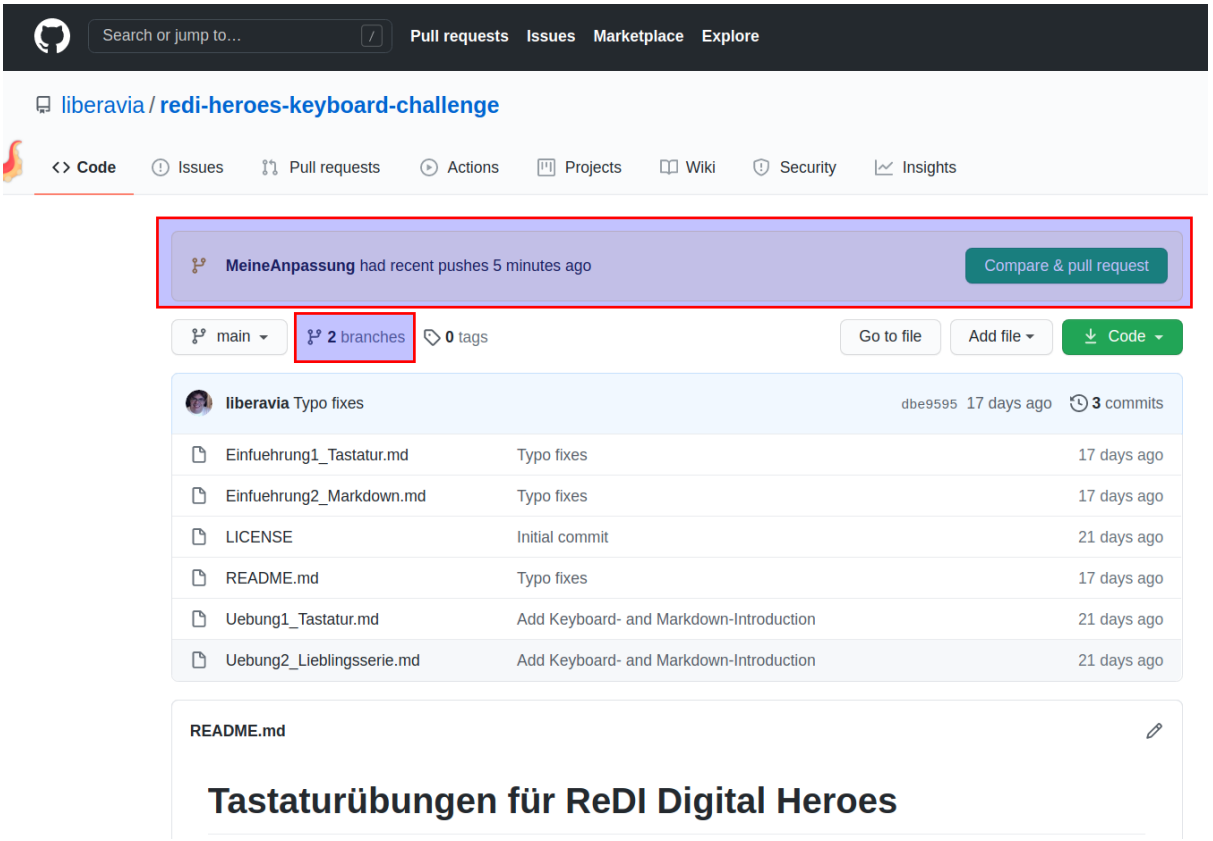
Wenn du der Meinung bist, dass du mit deinen **Änderungen fertig** bist, kannst du deinen **Branch auf GitHub hochladen**. Diesen Vorgang nennt man **Push** (engl. für schieben). Der Knopf dafür ist der grüne Pfeil nach rechts oben, der ebenfalls rechts oben in der IDE zu finden ist:



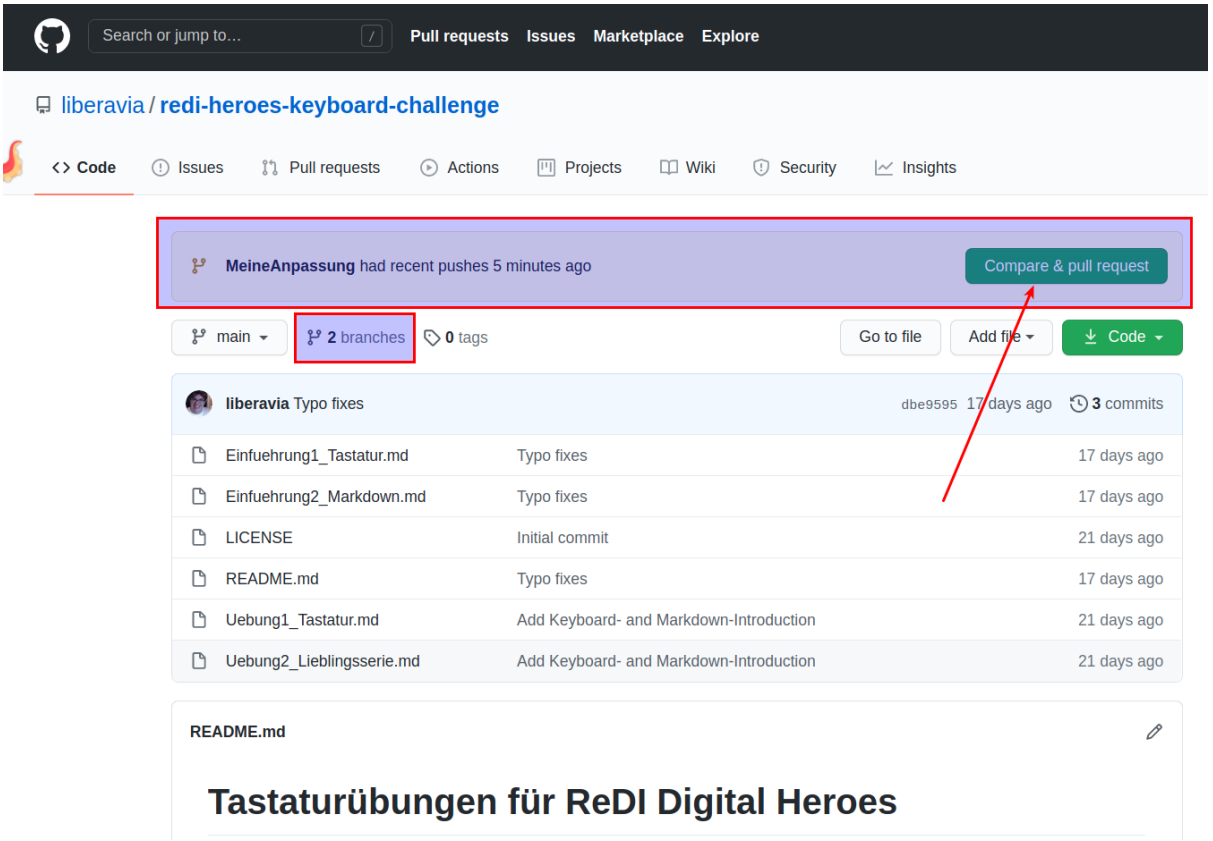
Nach dem Klick darauf erhältst du nochmal eine Zusammenfassung deiner Änderungen. Auf der linken Seite werden deine Commits aufgelistet und auf der rechten Seite alle geänderten Dateien.



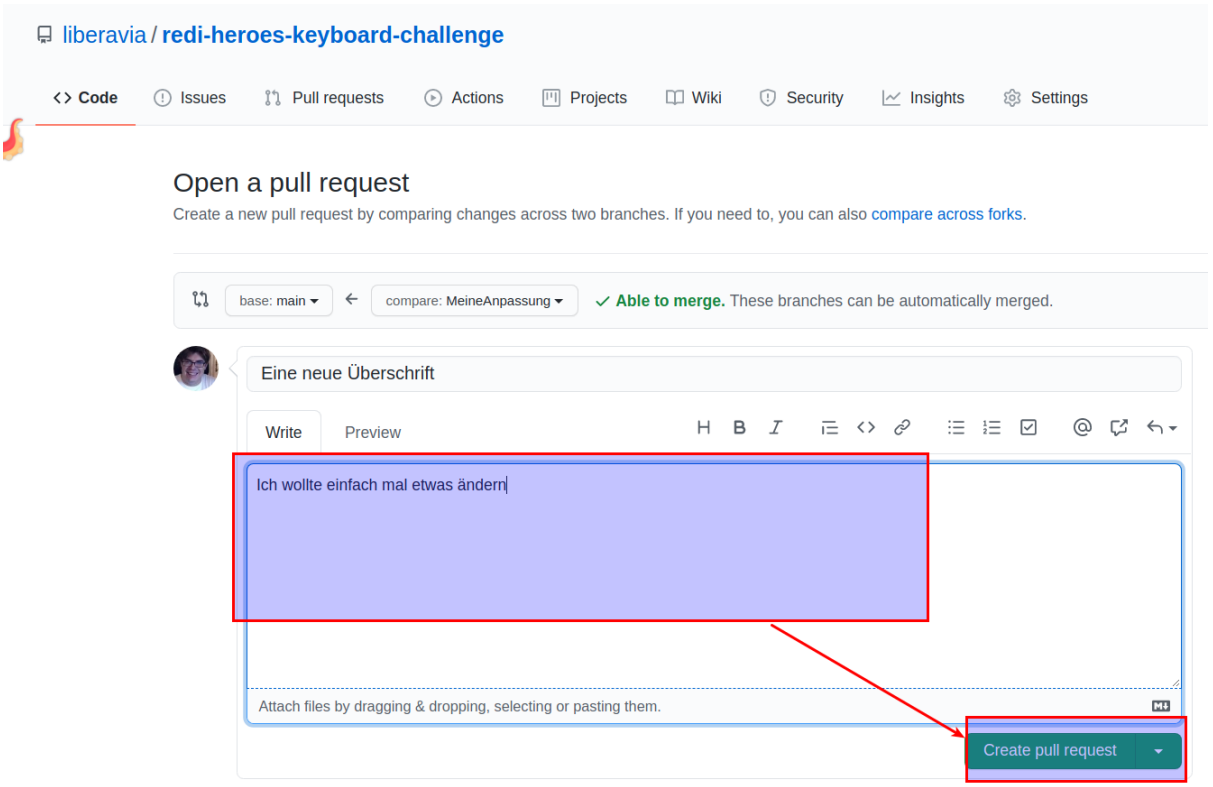

Wenn du jetzt in deinem GitHub-Konto nachsiehst, wirst du folgendes sehen:



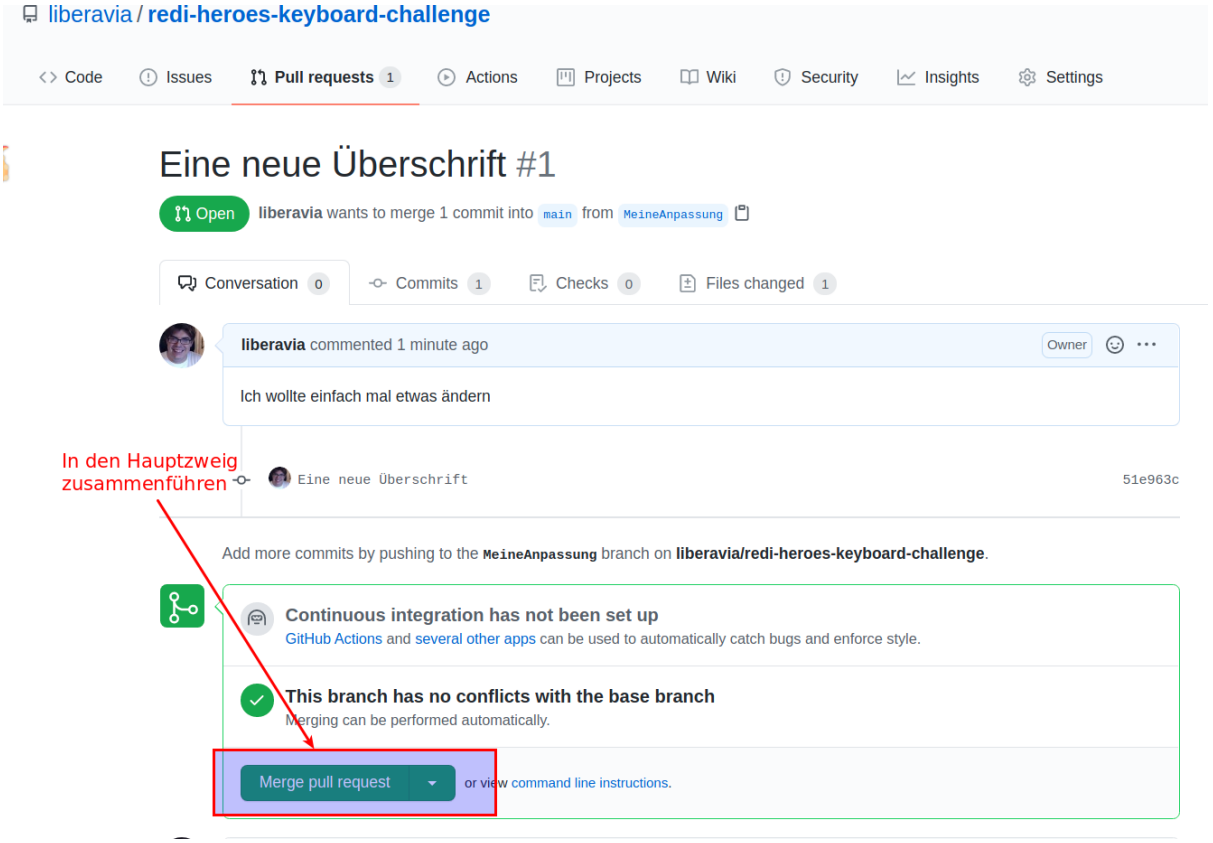

GitHub hat also registriert, dass du einen neuen Branch erstellt hast und bietet an einen sogenannten **Pull-Request** (engl. Anfrage die Änderung zu übernehmen) zu erstellen.



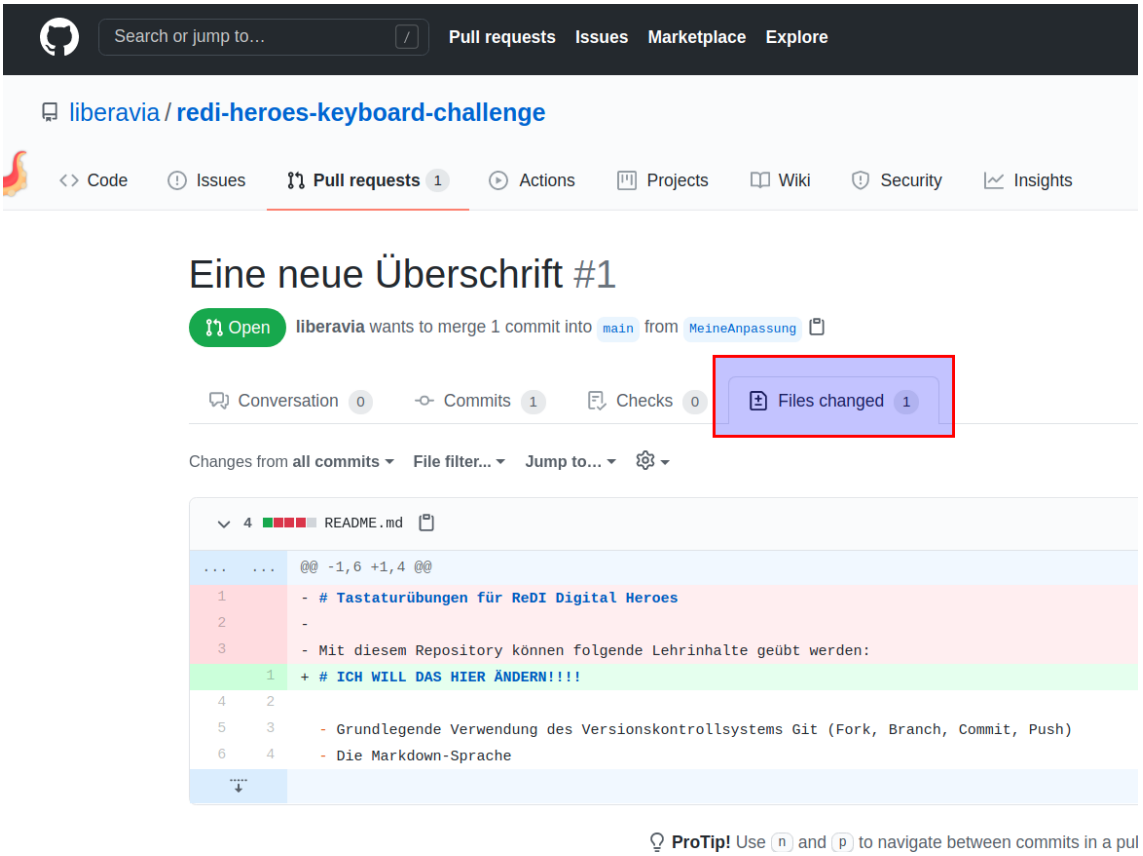

Ein Pull-Request dient dazu deine Anpassung in das Hauptprogramm zu integrieren. Es bietet die Möglichkeit sich mit den anderen Projektbeteiligten vorher auszutauschen.



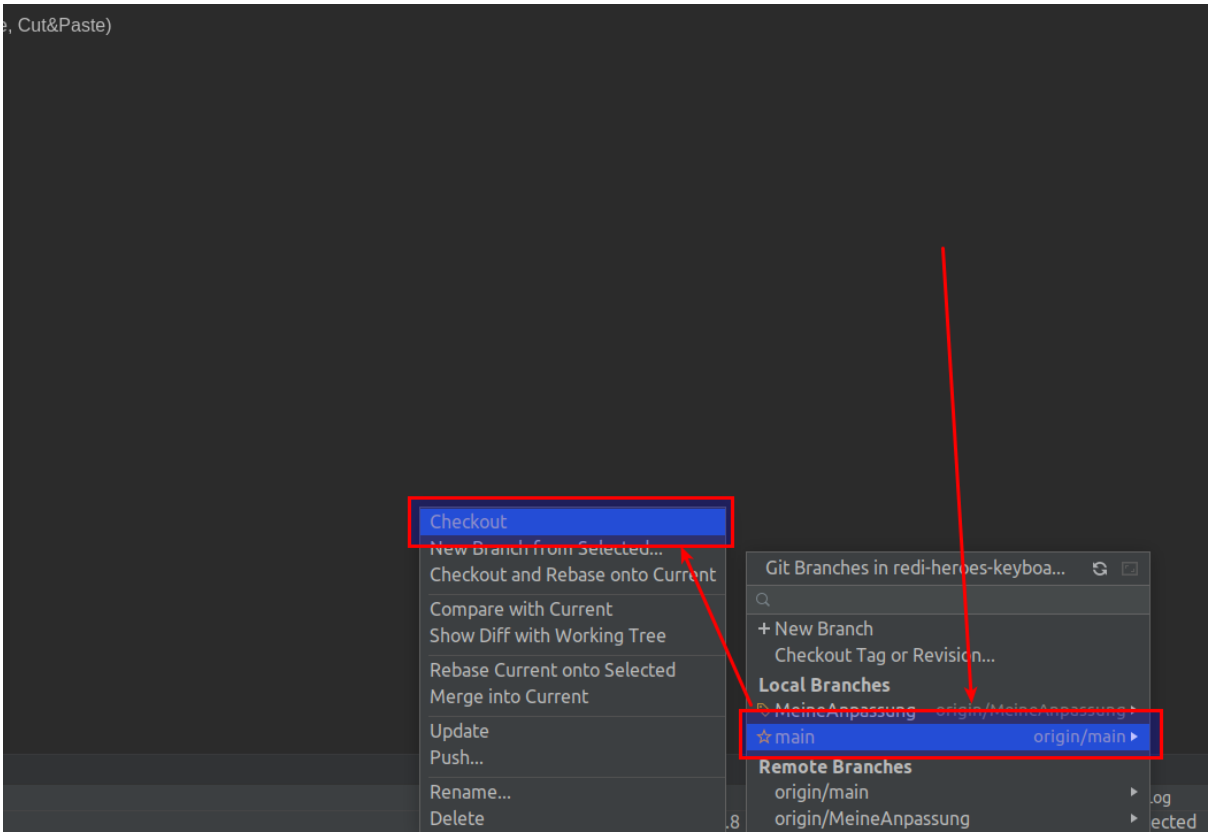
Ein Pull-Request lässt sich dann mit dem Hauptzweig zusammenführen.




In einem Pull-Request kann man auch genau sehen, was du geändert hast:



Wenn du deine Änderung dann in den Hauptzweig, dem *main-Branch* übernimmst, kannst du deinen Zweig im Anschluss löschen. Um nun von dem neuen Stand eine weitere Änderung durchzuführen, musst du den neuen Stand des main-Branch von GitHub erstmal in deine Kopie auf deinem Rechner übertragen. Dieser Vorgang nennt sich **Pull** (engl. ziehen).
Wechsel dazu in der IDE zurück auf den main-Branch:




Wenn du dann den blauen Pfeil, der nach links unten zeigt, klickst, werden die Änderungen dann auch bei dir aktualisiert. Von hier aus kann das ganze **dann von Vorne** beginnen.

**Der Vorgang in Kurz ist also immer:** *Checkout*, *Branch*, *Commit*, *Push*, *Pull-Request*, *Merge*.

Das ganze mag dir zunächst etwas kompliziert vorkommen, stellt aber sicher, dass jeder auf der Welt in deinem Projekt mitwirken kann und trotzdem alles nachvollziehbar und rückgängig machbar ist. Wir werden diesen Vorgang im Laufe des Kurses immer wiederholen, so dass es dir bald natürlich vorkommen wird und du am Ende des zweiten Kurses perfekt mit anderen im Team ein tolles Projekt baust.

# Computersprachen

Computersprachen werden dazu verwendet dem Computer Handlungsanweisungen zu geben, etwa das Lösen einer Mathe-Aufgabe oder die Anzeige eines Textfeldes.

# Es gibt viele Computersprachen

Es gibt etwa 700 Computersprachen (Zum Vergleich: Es gibt ca. 7000 Menschen-Sprachen). Alle haben ihren eigenen Einsatzzweck. Manche Computersprachen eignen sich eher für Mathe-Aufgaben, andere eher für schöne Darstellungen auf dem Bildschirm.
Die Sprachen die wir in unserem Kurs kennen lernen sind *Markdown*, *HTML* und *CSS*.

# Markdown - Computersprache zur Dokumentation

Markdown ist eine Sprache, die es ermöglicht schnell größere Dokumente zu erstellen. Entgegen eines Schreibprogramms wird das Aussehen des geschriebenen (Überschrift, Fettschrift etc.) über bestimmte Zeichen gesteuert.
Das klingt zunächst kompliziert, aber wenn du das einmal kannst (und das kann recht schnell gehen) bist du um ein vielfaches Schneller als mit einem Schreibprogramm.
Die wichtigsten Elemente der Sprache findest du hier:
https://github.com/liberavia/redi-heroes-keyboard-challenge/blob/main/Einfuehrung2_Markdown.md
Kurzlink zum eintippen: https://bit.ly/3ulOnGWv

# HTML - Computersprache zur Webseitendarstellung

Die *Hypertext Markup Language* (kurz: HTML) ist eine Computersprache die von Web**servern** gesprochen und von Web**clients** (meist Browser) verstanden wird. Sie ist eine sogenannte **Auszeichnungssprache** (Markup). Ihr Hauptbestandteil sind Tags.

## Was ist ein HTML-Tag?

Ein Tag (engl. für Etikett) steht für eine **bestimmte Anzeige-Funktion**. So steht der Tag “p” für einen Absatz (paragraph), der Tag “div” für einen Bereich (division). **Tags stehen in eckigen Klammern** und haben ein **Anfang und** ein **Ende.** Das Ende wird mit einem Schrägstrich gekennzeichnet. Zum Glück nimmt dir die IDE diese Arbeit ab und erstellt mit jedem Tag auch automatisch das Ende-Tag.

```
<p></p>
<div></div>
```

## Verschachtelung von HTML-Tags

Tags lassen sich beliebig ineinander bauen (verschachteln).

```
<div>
      <p>
            Ich bin ein Absatz in div
            <div>Ich bin ein div im Absatz von div</div>
      </p>
      <div>
```

```
            Ich bin ein div im div
            <p>Ich bin ein Absatz vom div im div</p>
      </div>
</div>
```

Das Ergebnis ist erstmal unspektakulär und sieht so aus:

Ich bin ein Absatz in div

Ich bin ein div im Absatz von div

Ich bin ein div im div

Ich bin ein Absatz vom div im div

**Tipp:** Wenn man in der IDE mit der Maus über einen Tag fährt wird automatisch auch dessen Gegenstück hervorgehoben.

## Eigenschaften (Attribute) von HTML-Tags

Tags können zusätzlich noch Eigenschaften haben, genannt: Attribute. **Attribute haben einen Namen und einen zugeordneten Wert**. Die **meistverwendeten Attribute** sind ***id*** und ***class***, wobei **id zur Identifikation** eines bestimmten Tags verwendet wird und der Wert **nur einmal** im gesamten HTML-Dokument vorkommen darf. In *class* dagegen dürfen mehrere Werte enthalten sein und diese dürfen auch häufiger vorkommen. **Attribute** werden **immer** innerhalb der spitzen Klammern **im Start-Tag** angegeben werden.

```
<div id="MichDarfEsNurEinmalGeben">
      <p class="rot fett">Ich bin ein fetter roter Absatz.</p>
      <p class="blau">Ich bin ein grüner Absatz.</p>
      <p class="blau unterstrichen">Ich bin blau und unterstrichen.</p>
</div>
```

Das Ergebnis ist nicht wie erwartet:

Ich bin ein fetter roter Absatz.

Ich bin ein grüner Absatz.

Ich bin blau und unterstrichen.

Wie die Eigenschaft class so genutzt wird, dass die obigen Dinge passen erfährst du im Kapitel zu CSS.

## Aufbau eines HTML-Dokuments

Ein HTML-Dokument ist stets gleich aufgebaut. Auch ein HTML-Dokument wird innerhalb eines Tags angelegt. Der Tag heißt “html”. Innerhalb dieses Tags gibt es **zwei Hauptbereiche *head* und *body***. Diese sind ebenfalls Tags. **In *head*** werden allgemein **unsichtbare Dinge** abgelegt, wie z. B. die Verbindung zu einer Datei, die Informationen über das Aussehen unseres Dokuments liefert (CSS). Ganz oben in der Datei wird noch ein spezielles Tag ohne ein Ende-Tag (<!DOCTYPE html>) verwendet, welches mitteilt, dass alles danach die Regeln befolgt, die wir gerade lernen:

```
<!DOCTYPE html>
<html>
<head>
      <title>Hier kommt der Titel für den Browser hinein</title>
</head>
<body>
    <h1>Ich bin eine Überschrift erster Ordnung!</h1>
    <p>Ich bin ein Absatz in body</p>
</body>
</html>
```

Hier ist die Ausgabe:

**Ich bin eine Überschrift erster Ordnung!**

Ich bin ein Absatz in body

## HTML-Tags für Überschriften, Absätze und Aufzählungen

Eine **Überschrift** wird **mit dem Tag <h> und einer zusätzlichen Nummer** angegeben. Die Nummer gibt dabei die Ordnung an von Hauptüberschrift zu Unterthemen:

```
<h1>Ich bin eine Hauptüberschrift erster Ordnung</h1>
<h2>Ich ein ein Unterthema der Hauptüberschrift</h2>
<h3>Ich bin ein Unterthema des Unterthemas</h3>
```

**Ich bin eine Hauptüberschrift erster Ordnung**

**Ich ein ein Unterthema der Hauptüberschrift**

**Ich bin ein Unterthema des Unterthemas**

das H im <h>-Tag steht für *heading* (engl. für Überschrift).

Absätze wurden ja schon häufiger gezeigt, aber hier nochmal zur Erinnerung. **Das Tag für den Absatz ist <p>,** Das p steht für paragraph (engl. für Absatz):

```
<p>Ich bin ein Absatz</p>
<p>Und ich bin der nächste Absatz</p>
```

Listen strukturieren Text sehr gut, wenn es um die Aufzählung von z. B. Fakten geht. Mit dem Tag <ul> erstellen wir eine solche unsortierte Liste (engl. unsorted list => ul). Die einzelnen Aufzählungspunkte werden mit dem <li>-Tag geschrieben.

```
<ul>
      <li>Listenpunkt 1</li>
      <li>Listenpunkt 2</li>
      <li>Listenpunkt 3</li>
</ul>
```

Das Ergebnis davon sieht so aus:

- Listenpunkt 1
- Listenpunkt 2
- Listenpunkt 3

**Tipp:** Teste mal das Tag <ol> anstelle des Tags <ul>

## Allzweck-HTML-Tags für Bereiche:  und 

Diese beiden Tags  und  haben keine eigene Funktion. Sie dienen dazu bestimmte Bereiche zu bestimmen um diese später zu stylen (Kapitel mit CSS). Du “spannst” sie sozusagen um bestimmte Bereiche. Stell es dir wie ein Gummiband vor.

Warum dann aber zwei Tags? Beide ähneln sich nur auf den ersten Blick. Der Unterschied liegt darin, dass ein ** -Bereich immer** mit einer **neue**n **Zeile** anfängt. Bei  ist das nicht so.

**Als Faustregel gilt daher:** Nur, wenn der zu umspannende (engl. to span) Bereich kein anderes Tag enthält, wird  verwendet. In allen anderen Fällen ist  die bessere Wahl.

Im folgenden Beispiel kannst du sehen, dass zwischen dem Anfang und dem Ende des -Tag sowohl der <h1>-Tag, als auch den <p>-Tag liegt. Dadurch kannst du später beides zusammen auswählen und zwar über das id-Attribut.

```
<!DOCTYPE html>
<html>
<head>
  <title>Titel der Seite</title>
</head>
<body>
```

```
  <div id="mein_bereich">
    <h1>Lieblingsfarbe</h1>
    <p>
      Meine Lieblingsfarbe ist <span class="rot">rot</span>
    </p>
  </div>
</body>
</html>
```

## Bilder mit <img>

Wenn du ein Bild auf deiner Webseite darstellen möchtest, dann steht hierfür das <img>-Tag bereit. Dieses Tag hat **ein Attribut** was **immer verwende**t werden muss: ***src***. Der Wert enthält den Ort, die Quelle (engl. source, Kurzform: src) von woher das Bild abgerufen werden soll.

Die Quelle kann dabei eine Internetadresse sein, aber auch eine heruntergeladene Datei, die an der gleichen Stelle (oder einem Unterordner) wie die HTML Seite liegt .

Mit einer Internetadresse, einem Link kannst du z. B. schreiben:

```
<img src="https://upload.wikimedia.org/wikipedia/en/0/02/Homer_Simpson_2006.png">
```

Wenn du im Internet ein Bild findest, kannst du die Adresse eines Bildes mit einem Rechtsklick auf das Bild selber kopieren:


Wenn du das später **für** eine **eigene Internetseite** tun willst, **achte bitte unbedingt darauf, dass du auch das Recht hast das ausgesuchte Bild zu verwenden**. Das ist leider

ziemlich kompliziert, daher solltest du hierzu unbedingt mit deinen Eltern darüber reden und Ihnen diesen Link geben:
https://www.ionos.de/digitalguide/websites/online-recht/bildrechte-im-netz/

Für deine Webseite **in unserem Unterricht kannst du alle altersgerechten Bilder verwenden**. Warum? Weil du die Bilder ja nicht veröffentlichst.

## Tabellen mit <table>,<tr> und <td>

Tabellen helfen dir dabei bestimmte Dinge übersichtlich darzustellen. Die verfügbaren Tags dazu sind:

- **<table>** markiert den **Bereich** in dem du eine Tabelle erstellen möchtest.
- **<tr>** steht für eine **Tabellenreihe** (engl. *table row*, Kurzform: tr).
- **<td>** steht für eine Spalte mit **Tabellendaten** (engl. *table data*, Kurzform: td). Hier stehen die eigentlichen Inhalte der Tabelle drin.

Ein Beispiel:

```
<table>
    <tr>
        <td>In Reihe 1, Spalte 1 steht etwas</td>
        <td>In Reihe 1, Spalte 2 etwas anderes</td>
    </tr>
    <tr>
        <td>Bei Reihe 2, Spalte 1 fängt es wieder an</td>
        <td>Ende Gelände bei Reihe 2, Spalte 2</td>
    </tr>
</table>
```

Ergebnis:

| | |
|---|---|
| In Reihe 1, Spalte 1 steht etwas | In Reihe 1, Spalte 2 etwas anderes |
| Bei Reihe 2, Spalte 1 fängt es wieder an | Ende Gelände bei Reihe 2, Spalte 2 |

Auch wenn die Tabelle noch keine Stilmittel verwendet (Linien, Fettschrift, Abstände), kannst du erkennen, dass die Spalten sauber untereinander dargestellt werden. Dazu kommen wir später im Kapitel CSS.

Um deine Tabelle noch besser zu gestalten, kannst du Spalten und Reihen auch mit einer Überschrift versehen. Für Überschriften verwendest du statt dem <td>-Tag das Tag <th> (engl. *table header*, Kurzform th). Das kannst du sowohl am Anfang der jeweiligen Reihe als auch der jeweiligen Spalte tun:

```
<table>
    <tr>
        <th></th>
        <th>Überschrift Spalte 1</th>
        <th>Überschrift Spalte 2</th>
    </tr>
    <tr>
        <th>Überschrift Reihe 1</th>
        <td>In Reihe 1, Spalte 1 steht etwas</td>
        <td>In Reihe 1, Spalte 2 etwas anderes</td>
    </tr>
    <tr>
        <th>Überschrift Reihe 2</th>
        <td>Bei Reihe 2, Spalte 1 fängt es wieder an</td>
        <td>Ende Gelände bei Reihe 2, Spalte 2</td>
    </tr>
</table>
```

Wie du sehen kannst, ist in dem <th>-Tag ganz am Anfang kein Inhalt. Das ist Ok, weil ein Inhalt hier keinen Sinn machen würde. Das Ergebnis von dem obigen Beispiel sieht dann so aus:

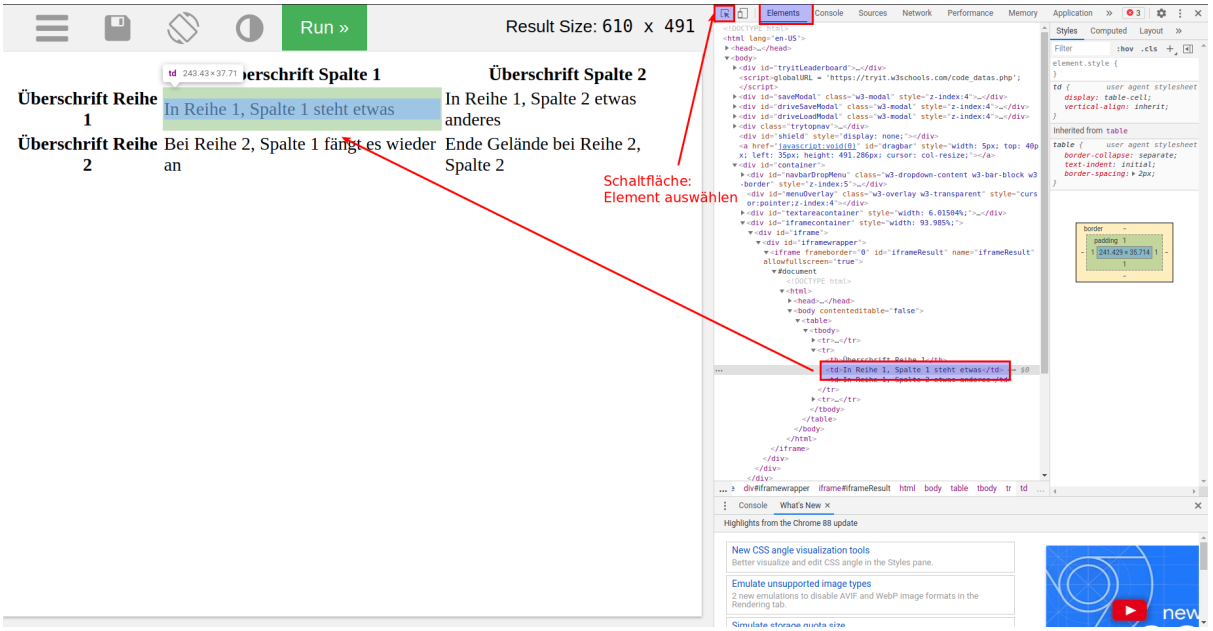
| | Überschrift Spalte 1 | Überschrift Spalte 2 |
|---|---|---|
| **Überschrift Reihe 1** | In Reihe 1, Spalte 1 steht etwas | In Reihe 1, Spalte 2 etwas anderes |
| **Überschrift Reihe 2** | Bei Reihe 2, Spalte 1 fängt es wieder an | Ende Gelände bei Reihe 2, Spalte 2 |

Wie gesagt: Mit der Computersprache CSS kannst du das aussehen der Tabelle noch wie gewünscht anpassen.

## Es gibt noch viel mehr HTML-Tags, Finde sie!

Sicherlich hast du dich schon gefragt, ob es noch HTML-Tags gibt, die dir bei der einen oder anderen Idee helfen können. In diesem Kurs können wir leider nicht alle behandeln. Es gibt z. B. Formular-Tags für Eingabefelder, Knöpfe usw. Diese behandeln wir im kommenden Kurs Web Developer Python.

Du kannst mit deinem Browser jedes Element auf jeder Webseite anzeigen lassen, indem du du die Taste F12 auf deiner Tastatur drückst. Das öffnet die Entwicklerwerkzeuge. Diese enthalten sehr viele Informationen zu der Webseite. Mit der Schaltfläche “Element auswählen” kannst du ein beliebiges Element auf der Webseite markieren und dir wird der Quellcode dieses Elements angezeigt:




Tags, die du nicht kennst, kannst du hier nachschlagen:

https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list
Link zum abtippen: https://mzl.la/3qP9xLJ

# CSS - Computersprache zur Verschönerung der Webseite

## Was bedeutet CSS und was kann man damit machen?

**CSS** steht für *Cascading Style Sheets* und lässt sich in etwa mit *Aufeinander aufbauende Stil-Anweisungen* übersetzen.
Diese Computersprache dient dazu das **Aussehen einer Oberfläche** zu **bestimmen**.
Du hast dabei eine riesige Auswahl von Formen, Farben und Schriften die du verwenden kannst.

## Wie sieht CSS aus?

Die Sprache CSS hat folgenden Aufbau:

```
Selektor {
      eigenschaft: wert;
      noch-eine-eigenschaft: wert;
}
```

Der Selektor (Auswähler) hat die Aufgabe zu zeigen für welchen Bereich auf der Webseite die Eigenschaften und Werte gelten sollen. In den geschweiften Klammern steht eine Liste

von Eigenschaften und Werten, die jeweils mit einem Semikolon voneinander getrennt werden.

Dabei gibt es viele Möglichkeiten den Selektor zu beschreiben. Wir konzentrieren uns im Kurs auf die drei wichtigsten:

1. Die Auswahl eines HTML-Tags, z. B. <h1>
2. Die Auswahl eines ID-Attributs
3. Die Auswahl einer Klasse

Ersteres ist recht simpel. Wenn du z. B. möchtest, dass alle **Überschriften** erster Ordnung in der Farbe **Lila** angezeigt werden sollen, dann sieht das in der Sprache CSS so aus:

```
h1 {
    color: purple;
}
```

Für die beiden anderen Fälle (id und Klasse), schauen dir am besten nochmal die HTML-Datei an, die wir für die Tags  und  verwendet haben:

```
<!DOCTYPE html>
<html>
<head>
  <title>Titel der Seite</title>
</head>
<body>
  <div id="mein_bereich">
    <h1>Lieblingsfarbe</h1>
    <p>
      Meine Lieblingsfarbe ist <span class="rot">rot</span>
    </p>
  </div>
</body>
</html>
```

In CSS-Sprache kannst du nun bestimmen, dass die Schriftfarbe in “mein_bereich” grün sein soll, aber der -Bereich mit der Klasse “rot” auch wirklich in roter Schrift angezeigt wird:

```
#mein_bereich {
      color: green;
}

.rot {
      color: red;
}
```

Eine **ID** wird demnach mit einer **Raute (Oder: Hashtag)** ausgewählt. Eine **Klasse mit** einem **Punkt**.

## Wie ist CSS mit HTML verbunden?

CSS kannst du zwar innerhalb der HTML Datei verwenden, aber es ist besser Aussehen und Darstellung der Seite direkt am Anfang voneinander zu trennen.
Hierzu dient der **<link>-Tag** (engl. *to link*, Verbinden), der **im <head>-Tag** der HTML-Datei die CSS-Datei einbindet:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Wie bei der *src*-Eigenschaft des <img>-Tags kann die Eigenschaft *href* (engl. Hyper Reference, Kurz: href) hier auch Dateien verwenden, die im gleichen Verzeichnis liegen (wie in dem Beispiel oben) oder es kann auch ein Internet-Link sein.

Im Kurs ist diese Verbindung schon gegeben. Ihr könnt euch voll darauf konzentrieren die ganzen eigenschaften auszuprobieren.

## Schriftart, -größe und -stil ändern

## Farben ändern

## Es gibt unglaublich viele Styles