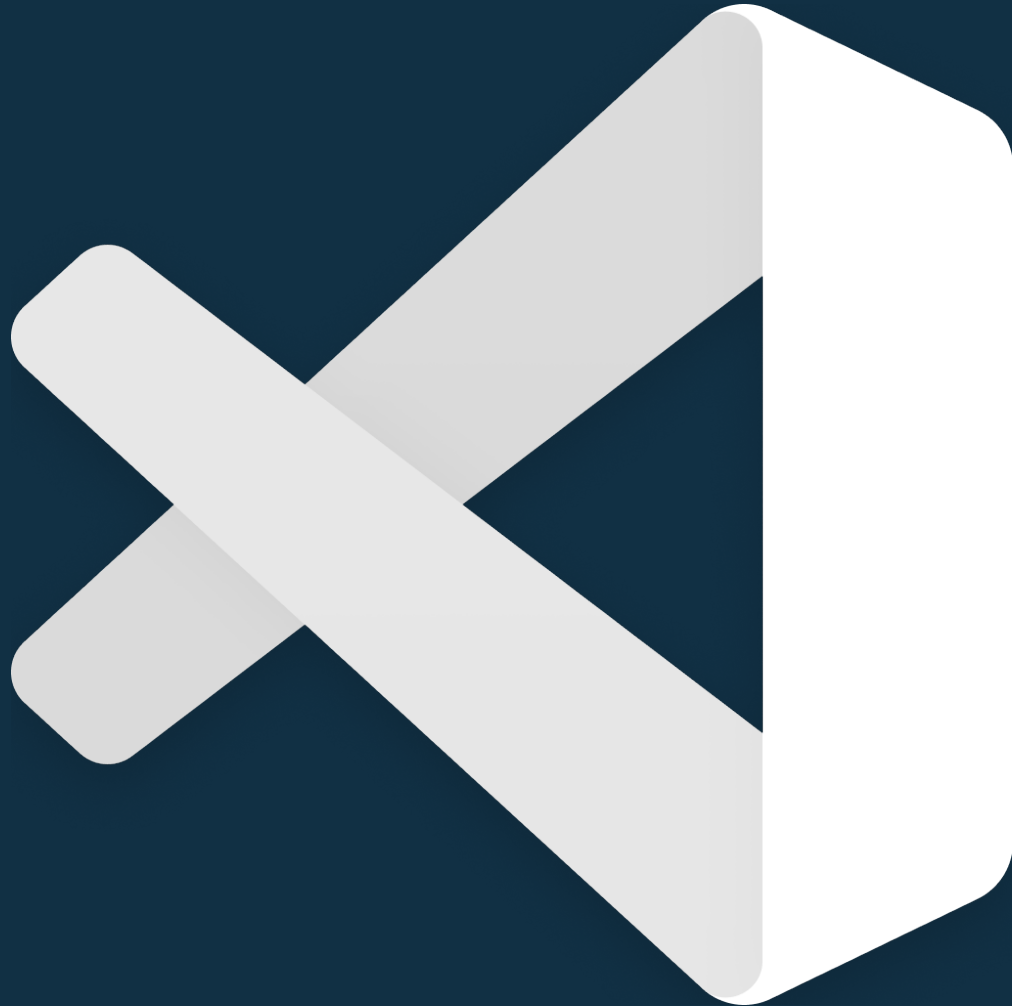


**CODING  
SCHULE**

# **CODING BASICS**

# AGENDA

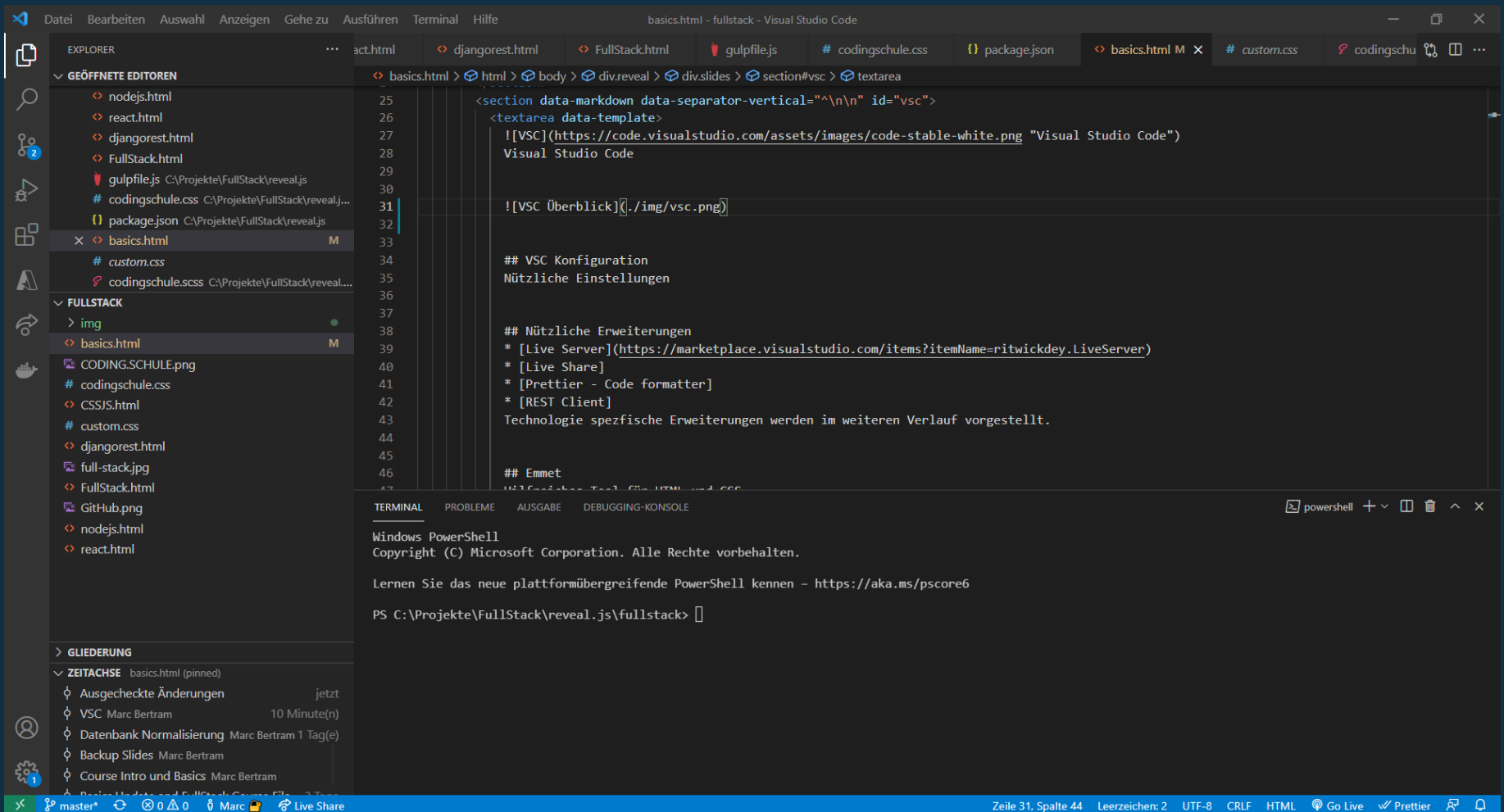
- Visual Studio Code
- git / GitHub
- REST APIs



Visual Studio Code

# VISUAL STUDIO CODE

- Quelltext Editor von Microsoft
- Plattformübergreifend für Windows, macOS und Linux verfügbar
  - Auch über den Browser nutzbar
- Unterstützung zahlreicher Programmier- und Auszeichnungssprachen
- Erweiterbar durch zahlreiche Plugins



<https://code.visualstudio.com/docs>

# TERMINAL

Im Abschnitt **Terminal** lassen sich Kommandozeilen aufrufen und darüber Befehle absetzen.

```
# Wechsel in einen Ordner
cd <Ordner>
# Wechsel in einen übergeordneten Ordner
cd ..
# Inhalt des aktuellen Ordners Anzeigen
ls
dir
# Verweis auf eine Datei/Ordner im aktuellen Ordner (./)
python .\datei.py
# Verweis auf eine Datei/Ordner im übergeordneten Ordner (../)
node ../ordner/datei.js
```

*In vielen Kommandozeilen können Eingaben durch drücken der **TAB-Taste** vervollständigt werden.*

# VSC KONFIGURATION

*Datei / Einstellungen / Einstellungen (STRG + ,)*

- Theme
- Schriftarten, Farbe, Größe, ...
- Automatisch Speichern
- Einstellungen von Erweiterungen
- Tastaturkürzel

Einstellungen lassen sich über einen Github oder Microsoft Account synchronisieren

# BEFEHLSPALETTE

Command Palette (*STRG* + *SHIFT* + *P*)

Ausführen von möglichen Funktionen aus einer Liste



## App.js - sports-trivia

JS App.js



keybindings.json

```
1  import React from 'react';
2  import {deepOrange500} from 'material-ui/styles/colors';
3  import AppBar from 'material-ui/AppBar';
4  import getMuiTheme from 'material-ui/styles/getMuiTheme';
5  import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
6
7  import QuestionBar from './QuestionBar';
8
9  const muiTheme = getMuiTheme({
10    palette: {
11      accent1Color: deepOrange500,
12    },
13  });
14
15  export default class App extends React.Component {
16    constructor(props, context) {
17      super(props, context);
18
19      this.state = {};
20    }
21  }
```

# EMMET

Vorinstallierte Erweiterung für HTML und CSS

<https://emmet.io/>

Kurzschreibweisen und Vorlagen

```
ul>li*3
```



```
<ul>  
  <li></li>  
  <li></li>  
  <li></li>  
</ul>
```

# NÜTZLICHE ERWEITERUNGEN

- [Live Server](#)
- [Live Share](#)
- [Prettier - Code formatter](#)

Viele weitere technologiespezifische Erweiterungen...

# LIVE SHARE

- Erweiterung für die gemeinsame Entwicklung in Echtzeit
- Gleichzeitiges Bearbeiten von Dateien
- Zugriff auf freigegebene Terminals und Server
- Live Share Audio

# LIVE SHARE DEMO



# AUFGABE



Passe die "About you"-Seite mit deinen Informationen an.  
Nutze hierfür Visual Studio Code.

[Vorlage](#)



# git

Verteilte Versionsverwaltung

- Freie Software zur verteilten Versionsverwaltung.
- Ursprünglich zur Verwaltung des Linux Kernels entwickelt (2005).

<https://git-scm.com/>



- Git dokumentiert alle Veränderungen an Dateien innerhalb eines Repositories.
- Dadurch kann parallel an einem Repository gearbeitet werden.
- Alle Änderungen können anschließend wieder zusammengeführt werden.

# GIT COMMANDS

Git ist ein Kommandozeilen Tool. Über die entsprechenden Befehle lassen sich neue Repositories anlegen, verändern, zusammenführen...

```
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-
        -p | --paginate | -P | --no-pager] [--no-replace-objects]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]
```

These are common Git commands used in various situations:

start a working area (see also: `git help tutorial`)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing

work on the current change (see also: `git help everyday`)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink



**GITHUB**

# GITHUB

- Cloudbasierter Dienst zur Versionsverwaltung auf Basis von **git**
- Verteilte Zusammenarbeit an Repositories und Bereitstellung über das Internet (Soziales Netzwerk)

[www.github.com](https://www.github.com)

Beliebte Alternative: [GitLab](https://gitlab.com)

# ÖFFENTLICHE GITHUB REPOSITORIES

- [React](#)
- [Visual Studio Code](#)
- [Python](#)
- [Corona Warn App](#)

# GITHUB DESKTOP

## Optional Client zur Verwaltung von GitHub Repositories

The screenshot displays the GitHub Desktop application window. The top menu bar includes 'File', 'Edit', 'View', 'Repository', 'Branch', and 'Help'. The main interface is divided into several sections:

- Current repository:** desktop
- Current branch:** esc-pr (with a green checkmark and commit #3972)
- Fetch origin:** Last fetched 2 minutes ago

The left sidebar shows a list of changes and history:

- Changes: Appease linter (iAmWillShepherd committed a day ago)
- History: Add event handler to dropdown compon... (iAmWillShepherd and Markus Olsson co...)
- Move escape behavior to correct compo... (iAmWillShepherd and Markus Olsson co...)
- Remove event handler from the branches.. (iAmWillShepherd and Markus Olsson co...)
- Merge branch 'master' into esc-pr (iAmWillShepherd committed a day ago)
- Merge pull request #4044 from desktop/... (Neha Batra committed a day ago)
- Merge pull request #4070 from desktop/... (Brendan Forster committed 2 days ago)
- bump to beta3 (Brendan Forster committed 2 days ago)
- Merge pull request #4057 from desktop/... (Brendan Forster committed 2 days ago)
- Merge pull request #4067 from desktop/... (Brendan Forster committed 2 days ago)
- Release to 1.1.0-beta2 (Neha Batra committed 2 days ago)

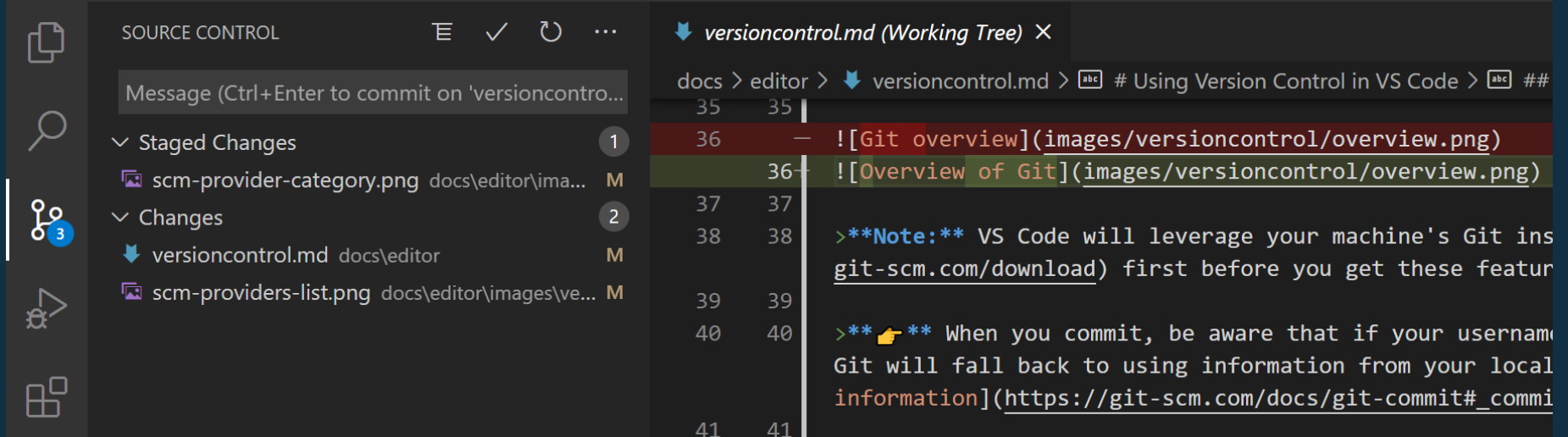
The main area shows a diff for the file `app\src\ui\toolbar\dropdown.tsx`. The commit message is "Add event handler to dropdown component" by iAmWillShepherd and Markus Olsson. The diff shows changes to the `ToolbarDropdown` component, including a new `isOpen` state and a new `onFoldoutKeyDown` event handler.

```
@@ -145,6 +145,10 @@ export class ToolbarDropdown extends React.Component<
  this.state = { clientRect: null }
}
+ private get isOpen() {
+   return this.props.dropdownState === 'open'
+ }
+
private dropdownIcon(state: DropdownState): OcticonSymbol {
  // @TODO: Remake triangle octicon in a 12px version,
  // right now it's scaled badly on normal dpi monitors.
@@ -249,6 +253,13 @@ export class ToolbarDropdown extends React.Component<
  }
}
+ private onFoldoutKeyDown = (event: React.KeyboardEvent<HTMLElement>) => {
+   if (!event.defaultPrevented && this.isOpen && event.key === 'Escape') {
+     event.preventDefault()
+   }
+ }
```

# GIT UND VSCODE

In Visual Studio Code sind bereits grafische Oberflächen zur Nutzung von git integriert.

Zusätzlich lassen sich zahlreiche Erweiterungen installieren, z.B. zur Integration von GitHub



# GIT WORKFLOW



Arbeiten mit git



# ANLEGEN EINES GIT REPOSITORIES

# ANLEGEN EINES GIT REPOSITORIES

Initialisieren eines neuen Repositories.

```
git init
```

# ANLEGEN EINES GIT REPOSITORIES

Initialisieren eines neuen Repositories.

```
git init
```

Hinzufügen von Dateien zum Commit

```
git add <file>
```

# ANLEGEN EINES GIT REPOSITORIES

Initialisieren eines neuen Repositories.

```
git init
```

Hinzufügen von Dateien zum Commit

```
git add <file>
```

Änderungen bestätigen: Commit

```
git commit
```

```
git commit -am "Beschreibung der Änderungen"  
# a = alle Dateien; m = Beschreibung
```

# AUFGABE



1. Installiere git auf deinem Computer
2. Initialisiere ein neues Repository
3. Füge Dateien hinzu und erzeuge verschiedenste Änderungen

Mit `git log` und `git diff` kannst du dir die protokollierten Änderungen angucken.

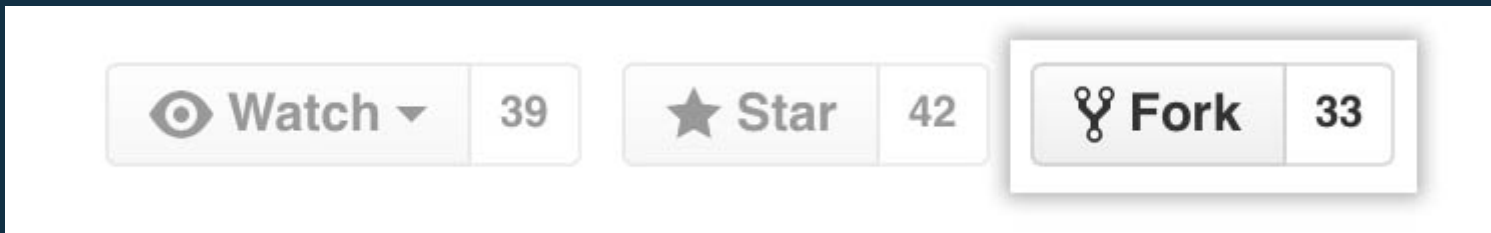
# GIT CLONE

Über `git clone` lassen sich Kopien eines vorhandenen Repositories (lokal oder remote) erzeugen. Jede Kopie kann unabhängig von einander bearbeitet und synchronisiert werden.

```
git clone https://github.com/Codingschule/course_profile.git
```

# GITHUB FORK

Clone eines GitHub Repositories in den eigenen GitHub Account.



# GIT SYNCHRONISIERUNG

git remote, git fetch, git merge, git push, git pull



# GIT REMOTE

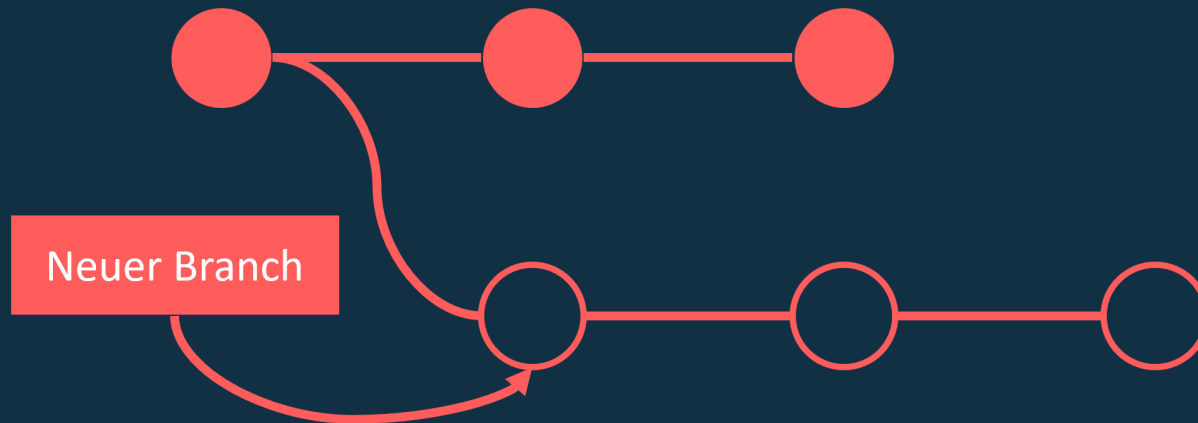
## Lesezeichen auf andere Repositories

```
$ git remote -v  
origin  https://github.com/Codingschule/fullstack-course.git (fetch)  
origin  https://github.com/Codingschule/fullstack-course.git (push)
```

Durch `git clone` werden automatisch Lesezeichen auf das ursprüngliche Repository (**origin**) gesetzt.

# BRANCHES

Über git branches können verschiedene Versionen (branches) deines Quellcodes parallel existieren. Der Standard Branch wird **master** oder **main** benannt.



# BRANCHES

Anzeigen aller lokalen Branches eines Repositories

```
git branch
```

Einen neuen Branch mit dem Namen **NextVersion** anlegen.

```
git branch NextVersion
```

Wechsel zwischen Branches

```
git checkout NextVersion
```

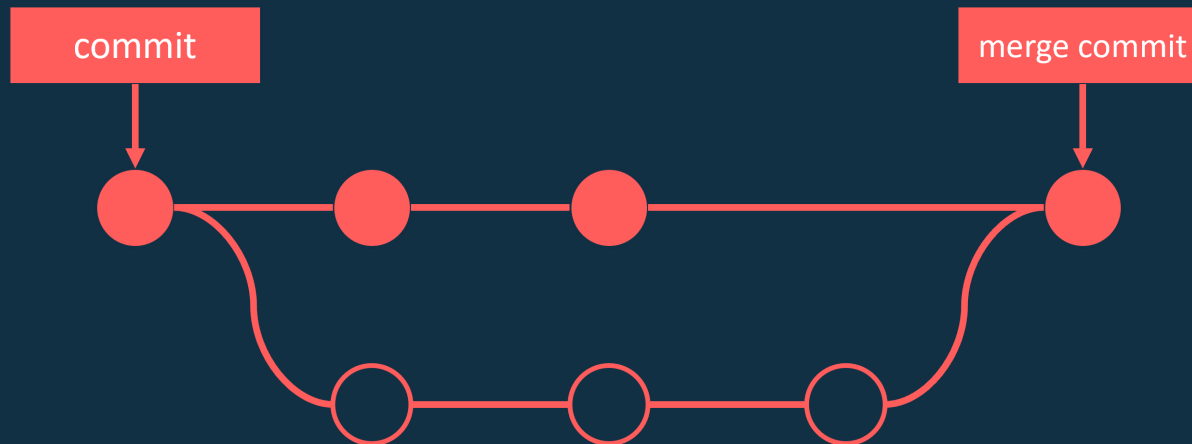
# GIT FETCH

Laden von Veränderungen aus dem Remote-Repository. Veränderungen werden jedoch nicht in deine lokale Umgebung übernommen. Es existieren, daher zwei verschiedene Versionen (Branches).

```
$ git fetch <remote>  
...  
$ git fetch origin
```

# GIT MERGE

Über den Befehl `git merge` lassen sich die Änderungen eines Branches mit einem anderen zusammenführen.



# GIT PULL

Kombination von `git fetch` und `git merge`.

```
$ git pull <remote>
...
$ git pull
# "git fetch origin" und "git merge"
```

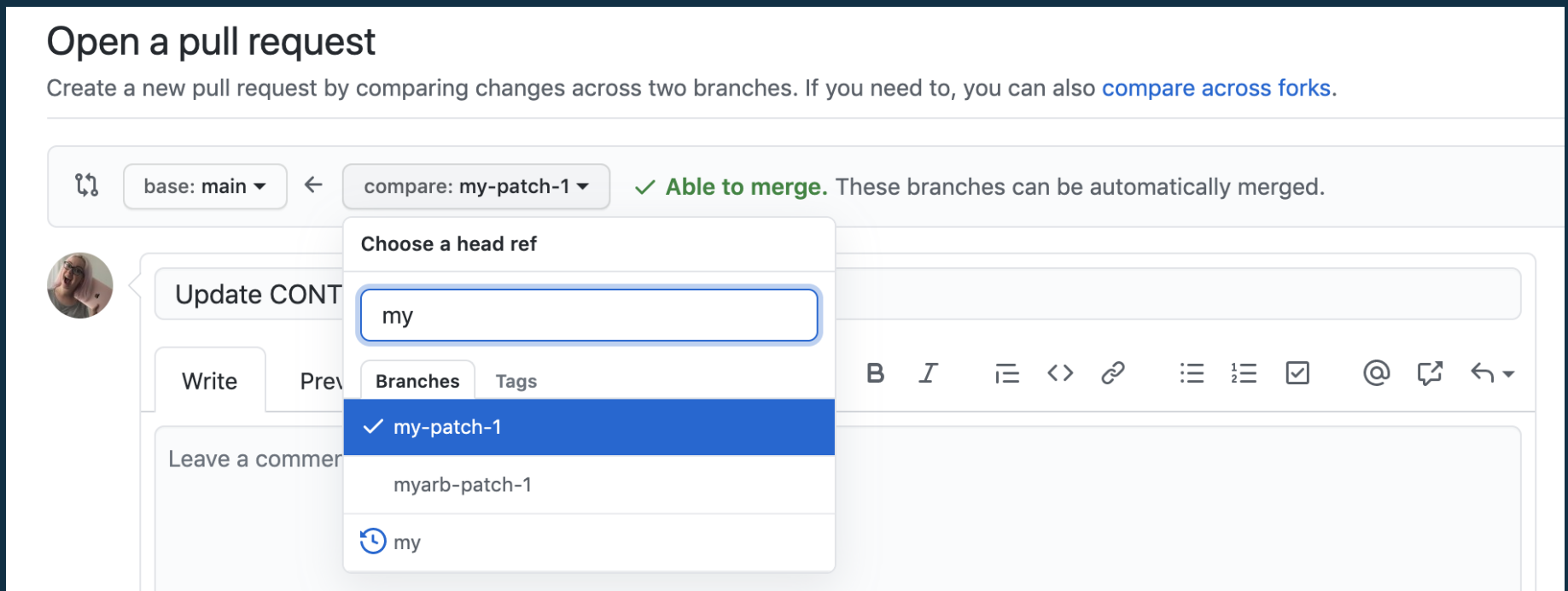
# GIT PUSH

Veröffentlichung von lokalen Änderungen im Remote-Repository

```
$ git push <remote> <branch>  
...  
$ git push origin main
```

# GITHUB PULL REQUEST

Änderungen an Branches lassen sich über GitHub mit einem anderen Branch zusammenführen. Über einen *pull request* können die Änderungen diskutiert werden und erst nach Freigabe zusammengeführt werden.



The screenshot shows the GitHub 'Open a pull request' page. At the top, it says 'Open a pull request' and 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).' Below this, there's a comparison bar showing 'base: main' and 'compare: my-patch-1'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' A dropdown menu is open, titled 'Choose a head ref', showing a search bar with 'my' and a list of branches: 'my-patch-1' (selected with a checkmark), 'myarb-patch-1', and 'my' (with a clock icon). The background shows a comment section with a profile picture and the text 'Update CONT...', 'Write', 'Prev', and 'Leave a comment'.



# GITIGNORE

Bestimmte Dateien und Ordner lassen sich von der Versionierung mit git ausschließen. Diese Ausnahmen können über die Datei `.gitignore` konfiguriert werden.

[.gitignore Vorlagen](#)

# GIT CHEAT SHEET UND TUTORIAL

[GitHub Cheat Sheet](#)

[git Tutorial](#)

# AUFGABE



- Kopiere das Profile Card Repository (fork) in deinen GitHub Account und lade den Fork über `git clone` auf deinen lokalen Computer.
- Erzeuge einen neuen Branch für deine Änderungen an der Profile Card und passe dort die Informationen an.
- Übertrage die Änderungen mit "push" in deinen GitHub Account.

Aufgabenstellung



# REST API

& Netzwerk Basics

# REST

## Representational State Transfer

- Architektur für den Datenaustausch zwischen Client und Server
- Konzentration auf Ressourcen und Zustandslosigkeit
- Realisierung erfolgt üblicherweise über das HTTP-Protokoll

# API

## Application Programming Interface

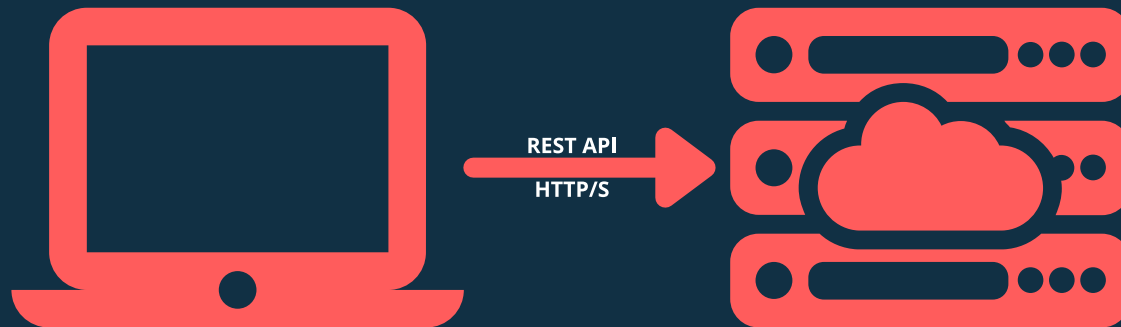
Schnittstelle zur Kommunikation von zwei Softwaresystemen untereinander. Eine Anwendung stellt eine API bereit und eine andere kann Dienste der API benutzen.

### Vorteile:

- Trennung von Programmteilen
- Klare Schnittstellen zwischen Programmteilen
- Wiederverwendbarkeit

# API UND FULL STACK APPS

In den aktuell gängigsten Full Stack Architekturen erfolgt die Kommunikation zwischen Frontend und Backend über eine REST API Schnittstelle.



# HTTP

## Hypertext Transfer Protocol

- Zustandsloses Protokoll zur Übertragung von Daten
- Standardprotokoll für die Übertragung von Webseiten im World Wide Web.
- Eine verschlüsselte Übertragung wird auch mit **https** gekennzeichnet.

<https://www.codingschule.de>



# URL

## Uniform Resource Locator

Der Aufruf einer http-Ressource erfolgt über eine URL.

**https://www.codingschule.de/fullstack?query=test**



# HOSTNAME

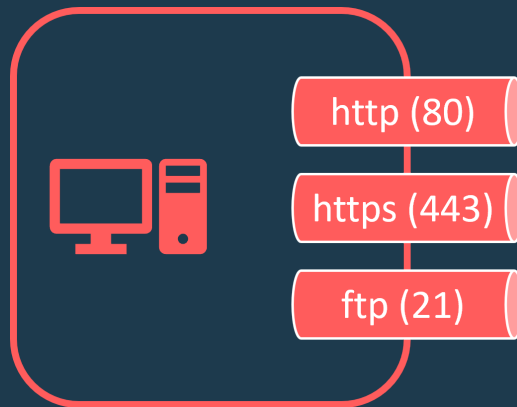
*Servername / IP-Adresse*

*Der lokale Computer wird auch als  
Localhost bezeichnet und ist über die IP-  
Adresse 127.0.0.1 erreichbar.*

# URL PORT

Hinter dem Hostname kann ein Port stehen.  
Ohne Port, wird der Standardport des Protokolls verwendet.  
Über verschiedenen Ports können mehrere Dienste auf einem Computer parallel bereitgestellt werden.

*http://localhost:3000*



# AUFGABE



Die **Cat API** gibt bei jedem Aufruf ein zufälliges Katzenbild zurück.

Der Aufruf der API erfolgt über die folgende URL:

<https://thatcopy.pw/catapi/rest/>

Rufe die URL auf und versuche das Katzenbild aus der Rückmeldung anzuzeigen.

# JSON

JavaScript Object Notation

Programmiersprachenunabhängiges Datenaustauschformat

```
{  
  "id":19,  
  "url":"https://thatcopy.github.io/catAPI/imgs/jpg/568c863.jpg",  
  "webpurl":"https://thatcopy.github.io/catAPI/imgs/webp/568c863.webp",  
  "x":56.94,  
  "y":50.91  
}
```

# HTTP PAKET

Bei Aufruf einer URL über HTTP werden viele verschiedenen Informationen in einem Paket übertragen.

Jedes Paket besteht aus einem **Header** und einem **Body**.

- Der Body enthält Nutzdaten der konkreten Anfrage.
- Im Header finden sich zahlreiche zusätzliche Informationen...

# HTTP BODY

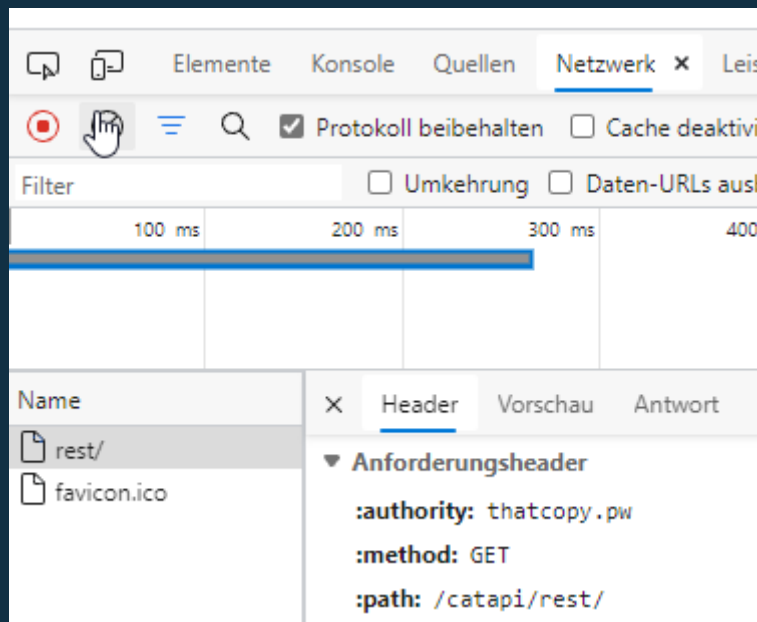
Im Kontext von APIs enthält der Body üblicherweise Daten im **JSON** Format.

Neben der Antwort kann auch die Anfrage (Request) JSON Daten beinhalten.

```
{  
  "name" : "Ada",  
  "text" : "Mein neuer Blog Eintrag"  
}
```

# HTTP HEADER

Über die Brower Entwicklungstools / -werkzeuge (F12) lassen sich die vom Browser aufgerufenen Verbindungen, sowie die dazugehörigen HTTP-Header anzeigen.





# AUFGABE



Schaut euch die HTTP Header der CAT API Anfrage und Antwort an.

Gibt es Header Informationen die ihr interessant findet?



# HTTP HEADER

```
:authority: thatcopy.pw
:method: GET
:path: /catapi/rest/
:scheme: https
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/v
accept-encoding: gzip, deflate, br
accept-language: de,de-DE;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
cache-control: max-age=0
if-none-match: W/"9f-X8cVdPNGMEFqwjDVMTAtKFNhWbM"
sec-ch-ua: "Chromium";v="94", "Microsoft Edge";v="94", ";Not A Brand"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
sec-fetch-dest: document
sec-fetch-mode: navigate
sec-fetch-site: none
sec-fetch-user: ?1
```

# HTTP METHODEN

HTTP kennt verschieden Methoden des Zugriffs.  
Die Art der Methode soll die Aktion bestimmen.

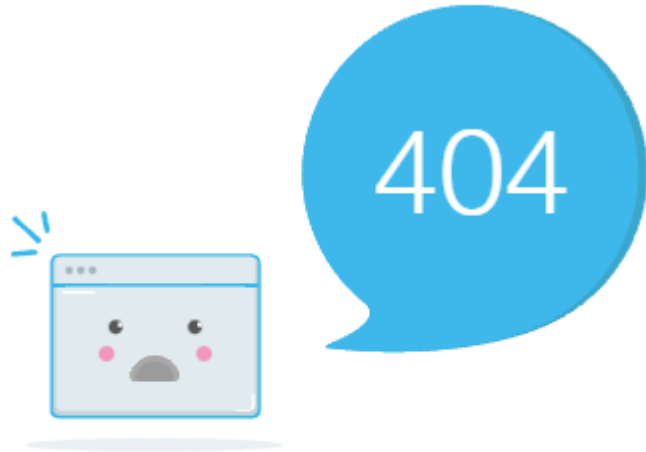
- GET - Abfragen von Daten
- POST - Erzeugen von Daten
- PUT - Überschreiben von Daten
- PATCH - Veränderung von Daten
- DELETE - Löschen von Daten
- HEAD, CONNECT, OPTIONS und TRACE

# AUFGABE



## HTTP METHODEN

Findet praktische Beispiele für die unterschiedlichen HTTP-Methoden bei einer Webapp/Webseite eurer Wahl.



SEITE NICHT GEFUNDEN

**HTTP STATUS CODES**

# HTTP STATUS CODES

Eine HTTP Antwort kann unterschiedliche Status Codes haben.

Der Status Code gibt einen Hinweis, ob die Anfrage erfolgreich bearbeitet wurde.

[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

# ÖFFENTLICHE APIS

- [Open Weather Map](#)
- [Currency Converter](#)
- [Marvel](#)
- [API-Football](#)
- [NASA API](#)
- [Yahoo Finance API](#)
- [Open Food Facts](#)
- [Public APIs Sammlung](#)

# AUTHENTIFIZIERUNG UND APIS

Viele APIs erfordern eine Authentifizierung beim Zugriff. Eine gängige Methode im Bereich der Webentwicklung ist die Nutzung von **Token**. Diese Token werden im Header der Anforderung übertragen.

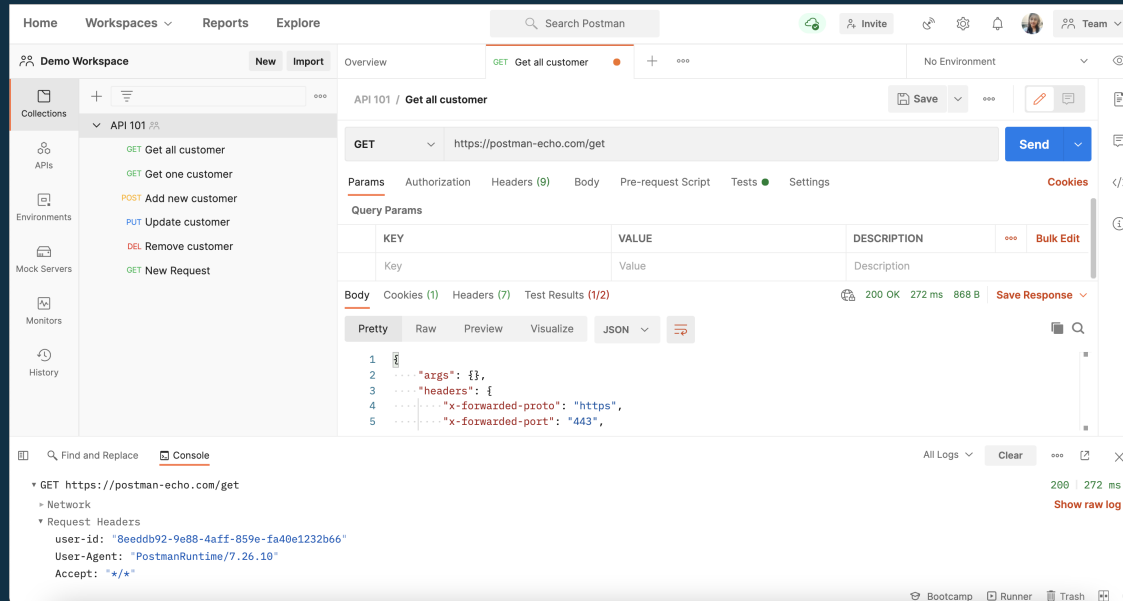
```
GET /api/user/ HTTP/1.1
Host: social-project-cs.azurewebsites.net
Connection: keep-alive
sec-ch-ua: "Chromium";v="94", "Microsoft Edge";v="94", ";Not A Brand"
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1b1
...
```



# POSTMAN REST CLIENT

Client zur Interaktion mit REST-APIs

Grafische Oberfläche zur Generierung von HTTP-Paketen



<https://www.postman.com/product/rest-client/>

# CURL

Kommandozeilentool zur Übertragung von Dateien

Auch für den Aufruf von APIs nutzbar.

Häufiges Beispiel in API Dokumentationen.

[GitHub API Dokumentation](#)

# AUFGABE



Github API Übung Passe deine Profil Informationen über die  
GitHub API an.

Aufgabenstellung