Dokumentation Projekt

Wetter-APP

Fachinformatiker-Anwendungsentwicklung Modul Javascript

erstellt von: Dennis Banasch

Team: FI-AE 2023

Betreuerin: Jana Koch

Datum: 02.02.2024

Inhaltsverzeichnis

1. Projektbeschreibung

1.1. Motivation zur Wahl der Aufgabe

Ich habe diese Aufgabe gewählt, weil ich diese Aufgabe von den vorgegeben Aufgaben innerhalb einer Woche als machbar erhielt. Außerdem habe ich noch nie so etwas wie eine Wetter-APP programmiert, was einer der Dinge ist, die man angehender Programmierer auf jeden Fall mal gemacht haben sollte. Zudem ist eine Wetter-APP für den privaten Gebrauch auch relativ nützlich.

2. Coding

2.1. Allgemeiner Aufbau des Programms

Mein Code ist im allgemeinen so strukturiert, dass es erstmal eine Haupt-Javascript-Datei „main.js“ gibt. Dort werden die Funktionen definiert, die ausgeführt werden, wenn das Programm gestartet oder beendet wird. Gestartet wird das Programm, wenn der User auf den Button „Run Program“ klickt. Die Funktion „starteProgramm“ wird ausgeführt, wobei immer der Wert des Buttons ausgelesen wird. Standardmäßig ist er auf „0“ wird dann aber immer jeweils beim starten oder beenden des Programms geändert. Je nachdem was für einen Wert dieser Button hat, wird das Programm gestartet oder beendet. Zusätzlich werden auch Texte geändert etc.

Die Hauptdatei selbst ist allerdings sehr schmal gehalten. Das liegt daran, dass mein Programm in sogenannte Segmente aufgeteilt ist. Insgesamt gibt es fünf Segmente (programmZustandSegment, selectedInputSegment, wetterAppSegment, datenAnzeigenSegment, chartDatenAnzeigenSegmente). Diese Funktionen sind alle in den Dateien „segments_01.js“ und „segments_02.js“ im „segments“-Ordner wierderzufinden. Hier ein kurzer Überblick was die einzelnen Segmente machen:

- programmZustandSegment
    - Übergabe der Argumente an Funktion
        - Übergebene Argumente haben Auswirkung auf den Wert den Buttons, Text usw.
    - Diese Funktion wird sowohl beim Start, als auch beim Beenden des Programms ausgeführt
- selectedInputSegment
    - Ausführung nur beim Start des Programms
    - Überprüfung der Auswahl der Benutzers
        - Insbesondere die Überprüfung, ob Geolocation ausgewählt wurde oder einfach nur eine Stadt
            - Nutzer wählt Geolocation
                - Rückgabe eines Objekts mit dem Status „geocheck“
            - Nutzer wählt Stadt
                - Rückgabe eines Objekts mit dem Status „cityCheck“ + Stadtdaten (Längengrad, Breitengrad, Stadtname)
- wetterAppSegment
    - Ausführung nur beim Start des Programms
    - Erwartet Objekt der vorherigen Funktion als Argument
    - Aufruf des Wetter-Workers
        - Wetter-Worker liefert die Wetterdaten über eine fetch-Methode
            - Standardmäßig werden geliefert: Location-Daten, Current-Daten (für heutigen Tag), Daily-Daten (für heute und die nächsten sechs Tage)
    - Funktion gibt beim erfolgreichen Ankommen der Daten („weatherCheck“) die Daten an das nächste Segment weiter
- datenAnzeigenSegment

    - Diese Funktion wird sowohl beim Start, als auch beim Beenden des Programms ausgeführt
    - Hier werden die Daten bzw. das Objekt vom vorherigen Segment erwartet und auf seine Keys geprüft
        - Je nach Key werden bestimmte Daten formatiert, DOM-Elemente generiert, aufgerufen und Daten in die HTML reingeschrieben oder drangehangen
    - Diese Funktion bekommt Daten und sorgt nur dafür, dass die Daten entsprechend so formatiert werden, dass sie auf der Seite dargestellt werden können
    - Auf Grund der vielen Fälle ist dieses Segment das längste Segment
    - Falls Daily als Key des Datenobjekts exisitert
        - Liefert Rückgabe Wert an nächstes Segement, um Daten in ein Chart zu konvetieren
- chartDatenAnzeigenElement
    - Falls das vorherige Segment einen Rückgabewert liefert, wird diese Funktion aufgerufen. Hier werden explizit die Daily-Daten des Datenobjekts an extrene Funktionen weitergegeben und dort so formatiert, dann sie in einem Chart (graphische Darstellung / Diagramm) angezeigt werden können

Zu der Segment-Struktur habe ich eine Klassenstrukur, um meine Funktionalitäten im Code zu gliedern. Ein Überblick über meine angelegten Klassen ist im Anhang vorhanden.

2.2. API-Connections

Ich greife in diesem Projekt auf zwei API's zu. Die erste API ist von opencagedata.com[1]. Diese API wird in meinem Code dazu verwendet, um den Ortsnamen meiner Geolocation auf Basis der übergebenen Längen- & Breitengraden des in HTML eingebauten navigator-Objekts zu erhalten. Diese API sorgt dafür, dass beim Auswahl der Geolocation bspw. „Max-Mustermann-Straße 45 Musterstadt" angezeigt wird. Wichtig bei dieser API ist, dass man für die Verwendung einen sogenannten API-KEY benötigt. Wenn man sich kostenlos auf der Seite registiriert, dann wird dieser Key zur Verfügung gestellt, sodass man die API auch nutzen kann. Entsprechende Anleitungen zum Verwenden der API sind auf der Webseite vorzufinden.

Die zweite API ist die eigentliche Wetter-API von open-meteo.com[2], welche die verschiedenen Wetterdaten basierend auf den übergebenen Längen- & Breitengrade anzeigt. Standardmäßig werden hier die Current-Daten & Daily-Daten abfragen, wobei hier nicht alle Parameter aus der API abgefragt werden. Die API stellt eine Seite zur Verfügung, wo man sich einzelne Links generieren lassen kann und entsprechende Parameter auswählen kann. Diese API bietet mit der Daily-Funktion die Möglichkeit sich die Wetterdaten von sieben Tagen anzeigen zulassen (heutiger Tag + die nächsten sechs Tage).

Wichtig noch zu wissen ist, dass beide API's immer darauf basieren, dass sie Längen-& Breitengrade als Daten erhalten. Diese Längen- & Breitengrad-Daten liegen für die einzelnen Städte im JSON-Format vor und sind von Chat-GPT[3] generiert worden. Dementsprechend ist die Richtigkeit dieser Daten nicht immer unbedingt gegeben. Die Längen- & Breitengrade der Geolocation werden über das navigator-Objekt des DOM's herholt, sodass entsprechende Google-Zugangsberechtigungen beim Aufruf der Seite akzeptiert werden müssen. Die erste API wird direkt beim Aufruf der Seite ausgeführt und entsprechende Daten bereits in eine GeoLocation-Object-Klasse reingeschrieben, während die zweite API erst beim Drücken des Button ausgeführt wird.

2.3. Erläuterungen zu der gewählten Notation

Meine Notationen habe ich grundsätzlich so nah wie möglich an der Camel-Notation gehalten. Dabei habe ich entweder deutschsprachige oder englischsprachige Namen verwendet. Die Namen sollen möglichst beschreibend für die Funktionalität sein. So habe ich allen Segmenten eine Segement-Endung gegeben. Außerdem habe ich eine Struktur beim Übergeben der Daten an einen Worker oder beim Empfangen von Daten eines Workers. Es ist grundsätzlich immer ein Objekt, welches übergeben wird mit folgender Struktur: {status: String, data: Any}. Diese Struktur kann abweichen., aber in der Regel übergebe ich so Daten, um eine bessere Struktur zu haben.

3. Fazit

3.1. Zusammenfassung

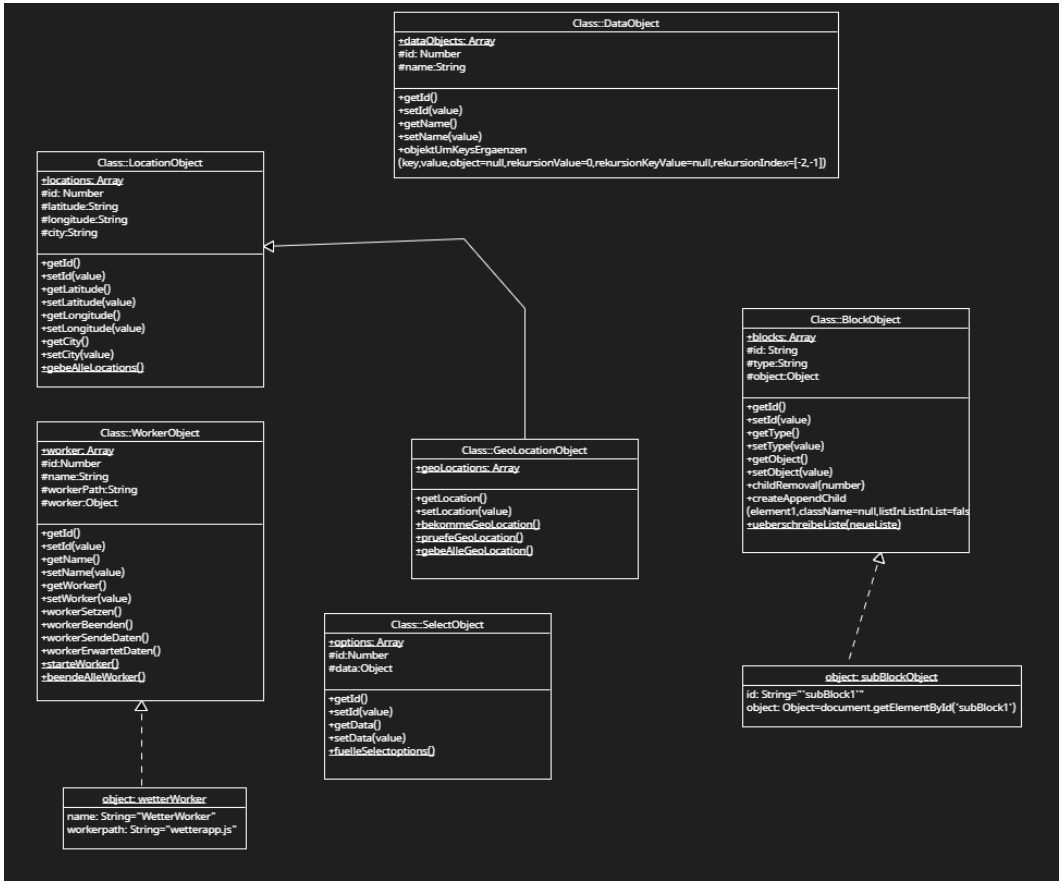
Das Projekt ist im Allgemeinen gut gelaufen und ich bin einigermaßen zufrieden mit dem Ergebnis, allerdings hat es auf Grund der Aufgabenstellung bezüglich Dokumentation usw. leider etwas an zeit gefehlt, sodass gewisse Features, die vorher geplant gewesen waren, wie etwa das Anzeigen der Wetterraten für vergangen Tage, nicht mehr eingebaut werden konnten. Insgesamt war es aber eine interessante Erfahrung mal so ein Projekt zu machen, da dies mein erstes richtiges JavaScript-Projekt war und ich Funktionalitäten verwendet habe, die ich vorher nie genutzt hatte (Worker-Object). Eine Wetter-APP zu programmieren ist eine der Aufgaben, die man als angehender Programmieren können sollte und ich bin stolz sagen zu können, dass ich für meinen ersten Versuch eine durchaus solide Leistung abgeliefert habe, jedoch hätte es mit mehr Zeit noch besser werden können. Meine Möglichkeiten waren durch den Zeitrahmen beschränkt. Der Zeitrahmen umfasst insgesamt eine Woche. Grafik zum Zeitstrahl in den Anlagen.

4. Quellen

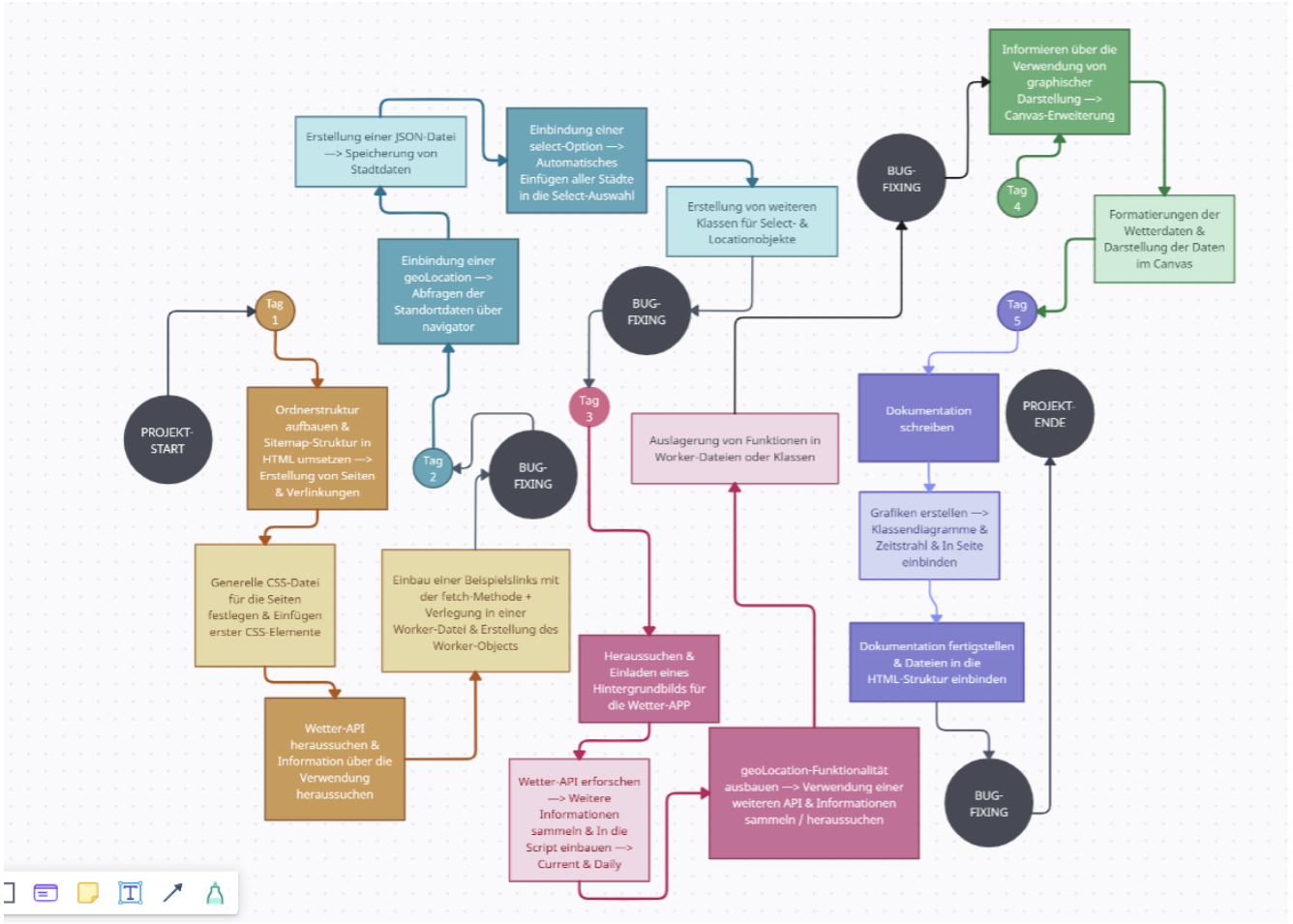
1. https://opencagedata.com
2. https://open-meteo.com
3. https://chat.openai.com
4. https://app.creately.com

5. Anlagen, Diagramme, vervollständigtes Sitemap

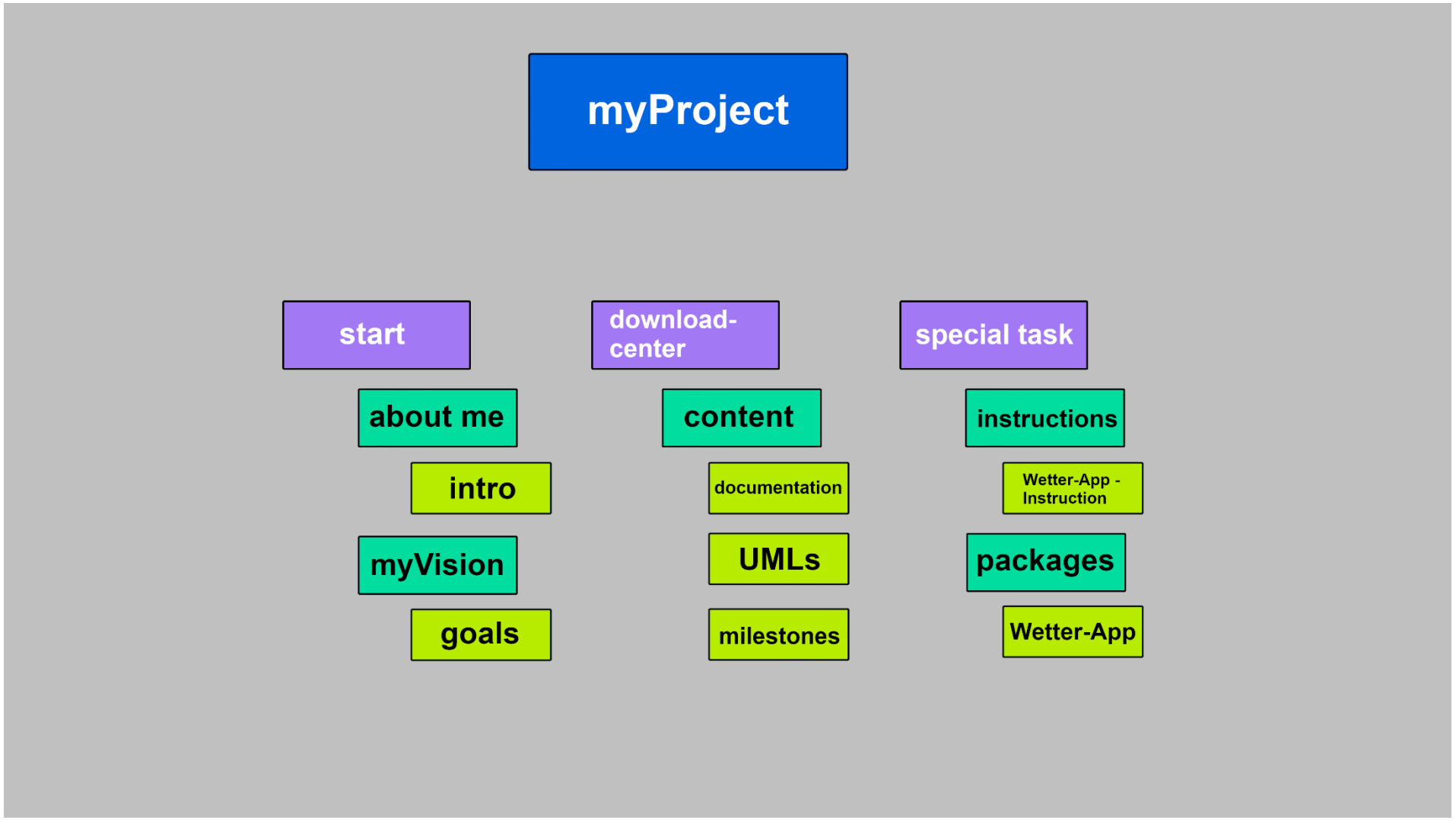
Klassendiagramm



Zeitstrahl


Erstellung einer JSON-Datei —> Speicherung von Stadtdaten
Einbindung einer select-Option —> Automatisches Einfügen aller Städte in die Select-Auswahl
Informieren über die Verwendung von graphischer Darstellung —> Canvas-Erweiterung
BUG-FIXING
Tag 4
Erstellung von weiteren Klassen für Select- & Locationobjekte
Formatierungen der Wetterdaten & Darstellung der Daten im Canvas
Einbindung einer geoLocation —> Abfragen der Standortdaten über navigator
BUG-FIXING
Tag 1
Tag 5
PROJEKT-START
Ordnerstruktur aufbauen & Sitemap-Struktur in HTML umsetzen —> Erstellung von Seiten & Verlinkungen
Tag 3
Auslagerung von Funktionen in Worker-Dateien oder Klassen
Dokumentation schreiben
PROJEKT-ENDE
Tag 2
BUG-FIXING
Grafiken erstellen —> Klassendiagramme & Zeitstrahl & In Seite einbinden
Generelle CSS-Datei für die Seiten festlegen & Einfügen erster CSS-Elemente
Einbau einer Beispielslinks mit der fetch-Methode + Verlegung in einer Worker-Datei & Erstellung des Worker-Objects
Heraussuchen & Einladen eines Hintergrundbilds für die Wetter-APP
Dokumentation fertigstellen & Dateien in die HTML-Struktur einbinden
Wetter-API heraussuchen & Information über die Verwendung heraussuchen
Wetter-API erforschen —> Weitere Informationen sammeln & In die Script einbauen —> Current & Daily
geoLocation-Funktionalität ausbauen —> Verwendung einer weiteren API & Informationen sammeln / heraussuchen
BUG-FIXING

Sitemap


myProject
start
about me
intro
myVision
goals
download-center
content
documentation
UMLs
milestones
special task
instructions
Wetter-App - Instruction
packages
Wetter-App