Praktischer Zwischentest – Gruppe B

Bei diesem praktischen Zwischentest gibt es 32 Punkte zu erreichen. Der Prozentanteil der erreichten Punkte von 32 (4 Extrapunkte möglich) trägt zu 36% zur Note in Web Scripting bei.

Allgemeine Hinweise:

- Der praktische Zwischentest muss auf Ihrem eigenen Rechner durchgeführt werden.
- Ziel ist es Ihre Umsetzungskompetenz mit JavaScript zu evaluieren
- Lesen Sie die Angabe genau durch
- Halten Sie Ihren Studierenden-Ausweis bereit

Unterlagen:

- Sie dürfen Internet, Bücher und eigene Programmstücke verwenden.
- Definitiv **verboten** ist die Unterstützung durch **andere Personen** bzw. ein **Datenaustausch** in jeglicher Form (manuell, elektronisch, …).
- Ihre Abgaben werden einer Plagiatsprüfung unterzogen.

Abgabemodus:

- Packen Sie Ihr gesamtes Projekt (alle HTML/JS/etc. Files) in **eine** zip-**Datei** und laden Sie diese in Moodle hoch.
- Dabei soll die vorgegebene Ordnerstruktur erhalten bleiben. Kontrollieren Sie Ihren erfolgreichen Upload bzw. die Validität der hochgeladenen Datei!

Wenn ein oder mehrere Ergebnisse zur Gänze oder auch nur in Teilen gleich oder sehr ähnlich sind (sodass ersichtlich ist, dass Ergebnisse kopiert wurden), dann wird die Prüfung für alle Beteiligten mit 0 Punkten beurteilt.
Die Verwendung von ChatGPT bzw. anderer KI-Systeme ist zur Lösung der Aufgabe untersagt und gilt als Erschleichung einer Prüfungsleistung.
Lesen Sie die Angaben genau und halten Sie sich an die vorgegebenen Nomenklaturen (Klassen, JS-Methoden, …)!

Aufgabenstellungen in grüner Schrift sind optional und können ggf. Punkte anderer fehlender bzw. unvollständiger Aufgabenstellungen kompensieren.

Viel Erfolg!

IMPLEMENTIERUNG WEBANWENDUNG

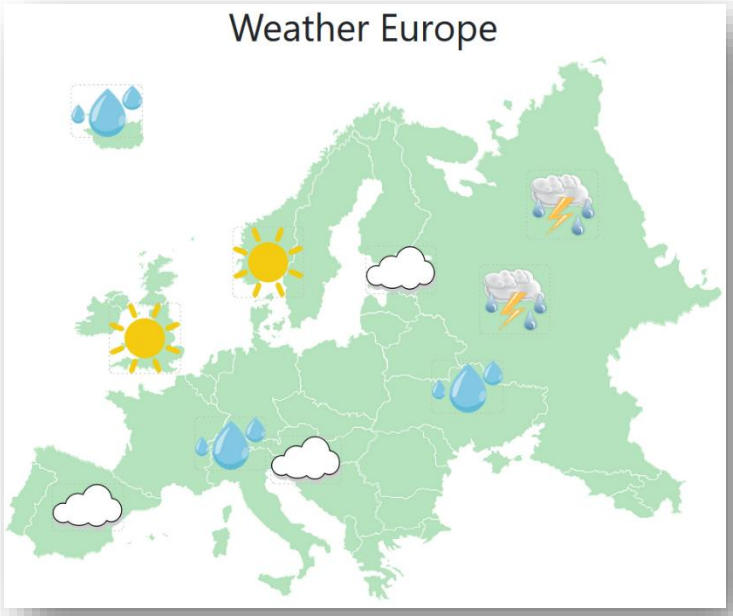
Vorbereitungen:

Nach dem Öffnen Ihrer Entwicklungsumgebung und dem Import der vorgegebenen Dateien erscheint die Projektstruktur, wie nachfolgend sichtbar. Dabei gilt, dass rote Teile erweitert werden müssen, schwarze Teile bleiben unverändert. Achtung! Sie müssen die Dateien über den Server ausführen, da sonst ein CORS Problem auftritt!

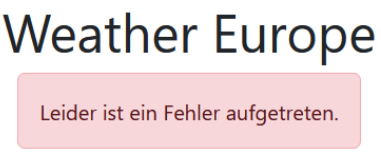
- Projektordner: **WeatherTemplate**
  - Datei: **index.html**
  - Datei: **getdata.js**
  - Datei: **myStyle.css**
  - Datei: **weatherdata.json**
  - Folder: **[img]** (beinhaltet Europakarte+WetterIcons)

Überblick Beispiel-Lösung:

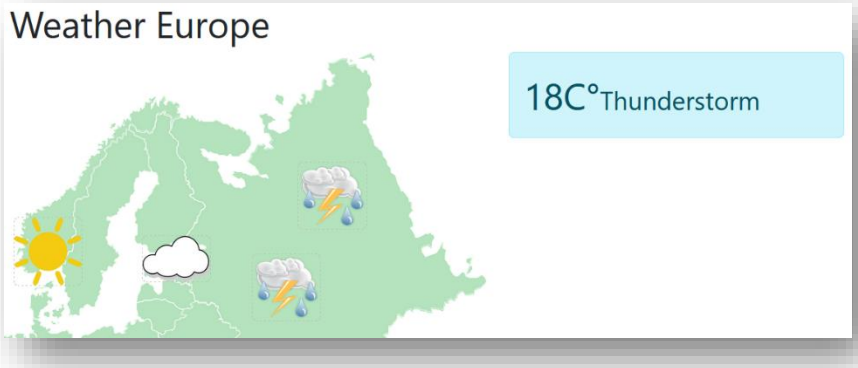
1) Beim Aufruf der Webseite wird Europa-Karte mit Wettericons automatisch geladen und wie folgt angezeigt:



2) Wenn der AJAX-Call nicht funktioniert, erscheint z.B. folgende Fehlermeldung:



3) Wenn einer der Icons aus **1)** mit der Maus angeklickt wird, werden die Temperatur- und Wetter-Informationen rechts neben der Karte angezeigt:



Tipp:
Arbeiten Sie am Styling erst nachdem Sie die Funktionalitäten implementiert haben.

1 Grundidee

*Auf der Webseite (**index.html**) werden die Wetterinformationen auf der Europa-Karte angezeigt, welche über die Datei **weatherdata.json** via JavaScript geladen werden. Die Wetter-Icons auf der Karte angeklickt werden. Daraufhin werden die Wetter- und Temperatur-Details zum Icon angezeigt.*

2 Beschreibung von *weatherdata.json*

*Bevor Sie mit der Entwicklung der Funktionalitäten beginnen, machen Sie sich bitte mit dem Inhalt der Datei **weatherdata.json** vertraut. Jedes Objekt in diesem Array enthält Information zu „weather“, „x“, “y“ und „degrees“.*

3 Umsetzung der Funktionalitäten in getdata.js

Setzen Sie folgende Funktionalitäten mit JavaScript um*:*

3a Binden Sie jQuery und das getdata.js-File in Ihr Projekt ein. **(2P)**

3b Implementieren Sie einen AJAX-Call der ausgeführt wird, wenn die Seite aufgerufen wird und fertig geladen ist. **(2P)**

i) Erstellen Sie eine Fehlermeldung, wenn die Daten nicht geladen werden können, in der Optik eines Bootstrap-Alerts. Im Falle eines Fehlers, wird keine Karte angezeigt. **(3P)**

3c Laden Sie die Inhalte von weatherdata.json via AJAX und nehmen Sie diese entgegen. **(2P)**

i) Hinweis: via `console.log(response);` können Sie die Daten auch in der Konsole sichtbar machen.

3d Für die Oberfläche werden die Daten aus dem JSON File benötigt.

i) Speichern Sie die nötigen Daten in Variablen und geben Sie diese in der Konsole aus. **(2P)**

3e Erstellen Sie auf Basis der Daten für jeden Eintrag ein Wettericon auf der Europakarte mit folgender Konfiguration:

i) Erzeugen Sie pro JSON-File-Eintrag ein Wettericon (Die Einträge entsprechen der Namen der png-Files). **(3P)**

ii) Positionieren Sie dieses via x/y-Koordinaten (css-Positionsangaben mittels top/left) auf der Karte (Europe). **(2P)**

iii) Weisen Sie jedem Icon für die korrekte Optik die Klasse „weather“ zu. **(1P)**

iv) Alle Icons sind per default nicht sichtbar und werden beim Laden der Site mit einem fadeIn von 1000ms animiert. **(3P)**

v) Zeigen Sie die Wetterbeschreibung sowie die Temperatur in der Spalte rechts neben der Karte in der Optik eines Bootstrap Info-Alerts an. **(3P)**

vi) Die Temperatur-Angabe soll in einer größeren Font-Size dargestellt werden. **(1P)**

vii) Sorgen Sie dafür, dass alle die Temperatur-Infos per Default ausgeblendet sind. **(2P)**

viii) Erstellen Sie einen Rahmen für die Wettericons via JavaScript. **(2P)**

3f Erstellen Sie eine neue Funktion mit dem Namen „*showTemperature()*“ welche aufgerufen wird, sobald ein Wettericon auf der Karte angeklickt wird. **(3P)**

3g Blenden Sie die nicht aktiven Temperatur-Infos mit einem SlideUp von 300ms aus und die aktive Information mittels SlideDown von 300ms ein - es soll immer nur die gerade angeklickte Temperatur-Info sichtbar sein. **(3P)**

3h Hinterlegen Sie den gerade angeklickten Wetter-Icon mit einer Farbe **(2P)**