universität innsbruck

Web mapping Projektarbeit

CliMAP Change

https://climapchange.github.io/index.html

Fakultät für Geo- und Atmosphärenwissenschaften am Institut für Geographie der Universität Innsbruck

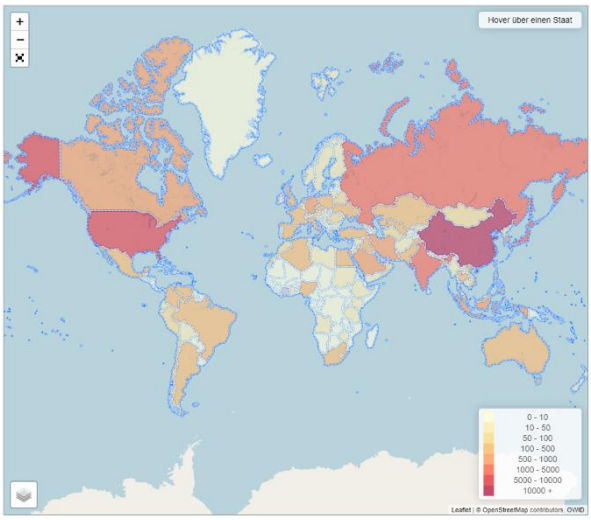
Kurs:	VU Geoinformatik: Web mapping
LV-Nummer:	716409
Betreuung unter:	Klaus Förster und Bernhard Öggl
Abgabetermin:	16.06.2021
Verfasst von:	Ann-Sophie Böhle (12046756); Sarah Rasi (01625401); Jens Weise (12028921)

Inhaltsverzeichnis

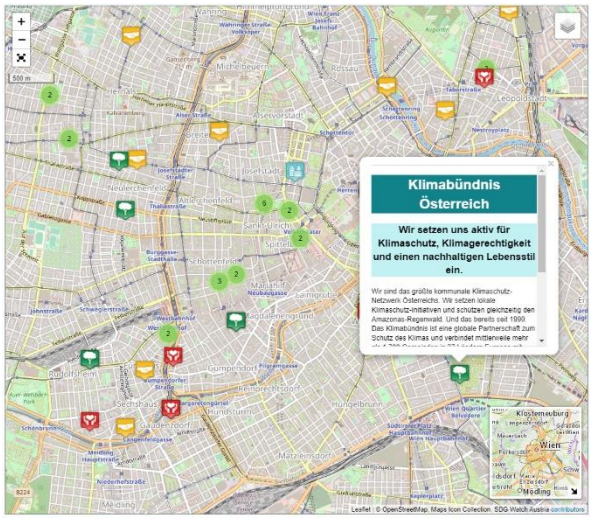
Allgemeiner Aufbau

Die Gruppe „CliMAP Change“ besteht aus Ann-Sophie Böhle, Sarah Rasi und Jens Weise. Da wir uns alle universitär und privat stark mit dem Klimawandel und Nachhaltigkeitsinitiativen beschäftigen, stand für uns schnell fest, dass wir in Projektarbeit diese Themen behandeln wollen. Durch die erstellte CliMAP Change Homepage und dazugehörige Karten wollen wir den Betrachter:innen die Thematik des Klimawandels, seine Folgen sowie Lösungsansätze zum Klimaschutz, anhand ausgewählter Daten näher bringen.

Es wurden drei Webseiten erstellt. Eine Startseite mit allgemeinen Informationen und einen Überblick, sowie Verlinkungen zu den andren zwei Webseiten mit den Karten. Karte 1 (unter „carbondata“) mit Informationen über die länderspezifischen Daten zu den CO_2-Emissionen. Karte 2 (unter „bestpractice“) zeigt die Mitgliedsorganisationen der SDG Watch Austria. Alle drei Webseiten, sollen ein einheitliches und simples Design haben. Die Startseite ist über: https://climapchange.github.io/index.html erreichbar.



Länderspezifische Emissions-Daten Karte
https://climapchange.github.io/carbondata/index.html



SDG Watch Austria – Mitgliedsorganisationen Karte
https://climapchange.github.io/bestpractice/index.html

Im Folgenden wird erläutert, wie die Webseiten und Karten erstellt wurden. Anhand von Screenshots aus dem Script werden die einzelnen Schritte visualisiert.

Hauptseite

Für die Hauptseite wurde eine index.html, und ein Ordner (mainsite) mit den verwendeten Bildern (Order: images) und eine style.css erstellt.

index.html

Im „head“ erfolgt zuerst die Verlinkung zu den verwendeten Daten und das Stylesheet. Beispielsweise wurde die „Raleway“ aus Google Fonts verwendet. Dafür wurde eine Verlinkung zur Googlefonts Seite geschaffen:

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CliMAP Change</title>

    <!-- Google Font Raleway-->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Raleway:wght@200&display=swap" rel="stylesheet">

    <!-- FontAwesome -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <link rel="stylesheet" href="/mainsite/style.css">
</head>
```

Im „body“ wird ein „header“ gebildet, wo ein einladendes Banner, der Titel und eine schnelle Navigation auf die anderen Seiten angezeigt werden. Für den Titel wird <h1> gewählt und im style.css formatiert. Darunter wird das Banner durch <img> eingebaut und die entsprechende Quelle (in <p>) dazu (mit <a>) verlinkt. Einige Verlinkungen werden in (durch taget=“_blank“ in <a>) in einem neuen Tab geöffnet. Durch das Element mit der ID „navigation“ werden die anderen zwei, von uns erstellten Webseiten verlinkt.

```
<body>
    <!-- Header mit: Titel, Bannerbild (+Bildquelle), schnelle Navigation zu Karten -->
    <header>
        <h1>CliMAP Change</h1>
        <img class="banner" width="1200" height="365" src="/mainsite/images/header2.jpg" alt="Eis im Meer">
        <p class="quelle"><a href="https://unsplash.com/photos/BWGsFbQheTQ" target="_blank">Bildquelle</a></p>
        <br>
        <div id="navigation">
            <a href="https://climapchange.github.io/carbondata/index.html" target="_blank"><i
                    style="margin-right: 0.5em"></i>Karte
                1</a>
            <a href="https://climapchange.github.io/bestpractice/index.html" target="_blank"><i
                    style="margin-right: 0.5em"></i>Karte
                2</a>
        </div>
    </header>
```

Das „main“ Element beinhaltet alle Texte und Informationen. Mit <h2> werden die Überschriften zu den jeweiligen Texten angegeben („Über uns“, „Über die Karten“, „Fakten“, „Was wir jetzt tun müssen“ und „Lust auf mehr? Weitere Informationen findest du hier:“). Textabschnitte <p> werden Klassen zugeordnet die dann im style.css individuell formatiert werden.
Bei „Über uns“ werden die jeweils zu unseren Namen, die github Accounts verlinkt. Es folgt eine kurze Einleitung zu uns und unserer Idee hinter CliMAP Change. Der github Account zu unserem Projekt wird ebenfalls verlinkt und mit einem Icon aus FontAwesome in einer Klasse im <i>-Tag eingebaut.

```
        Klimawandel.</p>
    <a href="https://github.com/climapchange/climapchange.github.io" target="_blank"><i class="fab fa-github mr-3"
            style="margin-right: 0.3em"></i>Unser github-Repo</a>
```

Bilder der Karten unter „Über die Karten“ sollen einladen es selbst auszuprobieren. Die Karten werden bei Erwähnung und auf den Bildern verlinkt und in einem neuen Tab geöffnet.

Als nächstes sind unter „Fakten“ kurze Informationen zum Klimawandel zu finden. Die dazugehörigen Quellen werden darunter verlinkt.
„Was wir jetzt tun müssen“ beinhaltet ein Video von Youtube, was direkt auf unserer Seite abgespielt werden kann.
Unter „Lust auf mehr? Weitere Informationen findest du hier:“ wurde mithilfe einer Auflistung, eine kleine Auswahl an weiteren Quellen gegeben.

```
<h2>Lust auf mehr? Weitere Informationen findest du hier:</h2>
<ul width="560">
    <li><a href="https://www.klimafakten.de/meldung/klimawandel-eine-faktenliste" target="_blank">Klimawandel
        Faktenliste</a></li>
    <li><a href="https://www.de-ipcc.de/media/content/SR1.5-SPM_de_181130.pdf" target="_blank">IPCC Bericht:
        1.5°C globale
        Erwärmung</a></li>
    <li><a href="https://www.klimafakten.de/meldung/du-und-der-klimawandel-viel-wissen-wenig-tun-die-infografik-zur-psychologie-des-handelns"
        target="_blank">
        Klimawandel: Viel wissen. Wenig tun? Die Psychologie es Handels</a></li>
    <li><a href="https://leitfaden.kommunaler-klimaschutz.de/" target="_blank">Praxisleitfaden zu kommumalem
        Klimaschutz</a>
    </li>
    <li><a href="https://www.umweltbundesamt.de/themen/klima-energie/klimafolgen-anpassung/werkzeuge-der-anpassung/tatenbank"
        target="_blank">Tatenbank: Projekte zur Anpassung an den Klimawandel</a></li>
    <li><a href="https://open.spotify.com/show/5j4zUNGyNT4MoRajQwTOQg?si=UbKFd00dRd-nx0psK6h-rA&dl_branch=1"
        target="_blank">Podcast: How to SDG!</a></li>
```

Im „footer“ folgt eine Verlinkung zur nächsten Seite und wird mit einem Pfeil am Seitenrand angedeutet. Für den Pfeil aus FontAwesome wird ein <i> Tag und der dazugehörige Klasse verwendet. Damit der Pfeil einen Abstand zum dazugehörigen Text bekommt, wird es mit 0.5em Abstand formatiert.

```
        </ul>
    </main>
    <footer>
        <a class="next" href="https://climapchange.github.io/carbondata/index.html"> hier gehts zur ersten Karte<i
            class="far fa-arrow-alt-circle-right next" style="margin-left: 0.5em"></i></a>
        <br>
    </footer>
</body>
```

Ein ähnlicher Aufbau mit gleichem Vorgehen im header und footer wird für „carbondata“ und „best-practice“ verfolgt.

style.css

Zunächst wird der wird der body formatiert. Die Ausrichtung soll automatisch geschehen (und die bereits erwähnte Schriftart soll „Raleway“ aus Google Fonts sein.
Die Überschrift „CliMAP Change“ im header unter <h1>, soll dick und weiß sein, ein Schriftgröße von 34px haben und durch einen leichten Schafften („text-shadow“) hervorgehoben werden. Zusätzlich sollen alle Überschriften eine Hintergrundfarbe bekommen, die farblich zu unserem Bannerbild passt.

```
header h1 {
    font-family: 'Raleway', sans-serif;
    font-size: 34px;
    text-shadow: 0 0 5px gray;
    color: #fff;
    background-color:#0a8992;
    font-weight: bold;
    text-align: center;
    margin: 16px;
}
```

Das header Image/Banner wird für alle Webseiten wie folgt formatiert:

```
header img{
    max-width: 95%;
    height:auto;
    width: auto;
    margin-top: 16px;
    margin-bottom: 16px;
}
```

Die links und besuchten Links und Verlinkungen sollen schwarz angezeigt werden, außer die Quelle im Banner (mit der Klasse „quelle“), dort soll es weiß.

```
a:link,
a:visited {
    color: black
}

.quelle,
.quelle a:link,
.quelle a:visited {
    color: #fff;
}
```

Teilüberschriften von <h2> sollen ebenfalls weiß sein, und die gleiche Hintergrundfarbe haben wie die <h1> Überschrift.
Die Textabschnitte werden oft der Klasse „text“ zugeordnet. Der Text soll bei Bedarf automatisch getrennt werden (hyphens: auto), einen Blocksatz haben (text-align: justfiy) und einen Abstand zu den rechten und linken Seitenrändern von 1.5em haben.

```
.text {
    hyphens: auto;
    text-align: justify;
    margin-right: 1.5em;
    margin-left: 1.5em;
}
```

Die Bildausschnitte der Karten mit der Klasse „pic“ sollen einen dünnen blauen Rahmen bekommen und das gleiche Format haben, wie das des Videos. Ebenfalls rundet es die Kanten des Umrisses des Bildes ab (durch „.pic:hover: border-radius“), damit die Nutzer*innen subtil vermittelt bekommen, dass sie beim Klick auf das Bild auf die Karte kommen.

```
.pic {
    max-width: 100%;
    height:315;
    width: 560;
    border: 2px solid #11828a;
    margin: auto;
}

.pic:hover {
    border-radius: 5rem;
}
```

Die Auflistung unter „Lust auf mehr? Weitere Informationen findest du hier:“ soll mittig ausgerichtet sein und daher auch die gleiche Weite wie das Video und die Bilder haben (560px).
Im footer wird das Pfeil Icon und der Text der Klasse „next“ für die Überleitung zur nächsten Seite an den rechten Seitenrand geschoben (float: right). Ebenso bekommt es eine blaue Farbe und einen Abstand zum rechten Seitenrand.

Als kleines ästhetisches Extra zum Hervorheben, bekommen alle Verlinkungen beim darüberfahren durch „a:hover“.

```
.next {
    float: right;
    size: 9x;
    margin-right: 1.0em;
    color:  rgba(16, 151, 185, 0.89)
}

a:hover {
    color: rgba(16, 151, 185, 0.89)
}
```

Mit „@media screen and (max-width:)” kann das Format für verschiedene Bildschrimgrößen angepasst werden.

Länderspezifische Emissions-Daten

Für die länderspezifische Emissions-Daten Karte wurde ein Ordner namens „carbondata“ erstellt, der alle verwendeten Dateien beinhaltet. Dasselbe vorgehen folgt für die Seite der SDG Watch Austria Karte.

index.html

Im „head“ erfolgt erneut zuerst die Verlinkung zu den verwendeten Daten und Skripten. Beispielsweise wurde eine Verlinkung zu FontAwesome, Leaflet und Leaflet Providers hergestellt.
Das Vorgehen im „header“ und „footer“ ist gleich, wie bei der index.html der Startseite. Im „main“ zeigt eine <h2> Überschrift, dass wir uns auf der Seite der ersten Karte befinden. Ein prägnanter Text, über die Features der Karte folgt. Darunter wird die Karte mit einem Element mit der ID „map“ implementiert.

```
<main>
    <h2>Karte 1</h2>
    <p class="text">Diese Karte gibt einen Überblick über die länderspezifischen CO<sub>2</sub>-
        In den verschiedenen Layern bekommst du einen schnellen Überblick, durch eine Farbrampe.
        die Möglichkeit die
        Höhe der CO<sub>2</sub>-Emissionen eines Landes einzusehen. Ein weiteres Layer liefert I
        die CO<sub>2</sub>-Emissionen pro Kopf in einem Land zu betrachtet. Im dritten Layer wir
        Anteil der CO<sub>2</sub>-Emissionen eines Landes am globalen Ausstoß angezeigt. Das rüc
        ganz andere Perspektive, oder nicht?</p>

    <p class="h4">Möchtest du genaueres wissen? <br>
        Zum Beispie, wie groß genau die Emissionen eines Landes pro Jahr sind? <br>
        Oder wie hoch die durchschnittlichen Pro-Kopf-Emissionen in Deutschland sind? <br>
        Oder wie groß der Anteil eines Landes an den globalen Emissionen ist? <br>
        <br> Kein Problem!
    </p>
    <p>Fahre ganz einfach mit der Maus über das Land und das Ergebniss wird dir angezeigt.</p>
    <div id="map"></div>
```

main.js

Die Funktionsweise der Karte wird über die Datei „main.js” definiert. Die interaktive Choroplethenkarte wurde dabei nach dem Beispiel eines Leaflet-Tutorials (Leaflet 2021) https://leafletjs.com/examples/choropleth/) entworfen. Im Gegensatz zu diesem können - abgebildet auf einem Baselayer, welcher über das Providers-Plugin bezogen wird - fünf Overlays dargestellt werden. Sie sind innerhalb des Layer-Control so eingebaut, dass jeweils nur ein Layer aktiviert sein kann. Innerhalb der Layer werden jeweils länderspezifische Informationen zu den Emissionen dargestellt. Die Daten wurden als JSON von Our World in Data (OWID 2021) bezogen und in der Datei „owid-co2-data.js” gespeichert.

Da die Daten innerhalb der Arrays verschachtelt sind, wurde eine Funktion entworfen, die auf die spezifischen Daten zugreifen kann. In der Funktion „getData” wird mithilfe des Parameters „polyName” zuerst die Array-Nr. des letzten gelisteten Jahres, für welches Daten vorhanden sind ermittelt und in der Variable „lastYear” gespeichert. Dafür wird die Länge der Array abgefragt und mit eins subtrahiert, da die Zählung bei null beginnt. Mithilfe der Variable kann je nach Information, die als Parameter „dataType” (geschrieben wie der „ key” innerhalb von „owid-co2-data.js”) gepasst wird, der entsprechende Wert für ein Land returnt werden.

```
// Funktion um CODATA-Daten abzurufen
function getData(polyName, dataType) {
    let lastYear = CODATA[0].country[polyName].data.length - 1;
    return CODATA[0].country[polyName].data[lastYear][dataType];
}
```

Bei den Daten, die auf diese Weise gewonnen werden, handelt es sich jedoch lediglich um Zahlenwerte. Wie werden diese mit Geodaten verknüpft? Dazu dient eine GeoJSON-Datei, die alle Länder der Welt als Polygone beinhaltet und von der Seite von Ash Kyd gedownloadet wurde. Diese kann ohne Namensnennung verwendet werden. Um gerade für die Smartphone-Nutzung Platz zu sparen, wird die Quelle (anders als bei OWID) deshalb nicht über die „attributionControl" hinzugefügt. Innerhalb der Datei „custom.geojson" befinden sich neben den Koordinaten der Ländergrenzen einige Metadaten, wie verschiedene Schreibweisen für ihre Namen. Anhand dieser Namen (teilweise war eine Anpassung notwendig) werden die Daten mit der Funktion „getData" gesucht. Gecallt wird die Funktion innerhalb der einzelnen Overlay-Style-Funktionen. Zur Bestimmung der „fillColor" wird dort der Zahlenwert mithilfe des Namens (Parameter „polyName") und der jeweils im Overlay abzubildenden Information (Parameter „dataType") ermittelt und dieser der jeweiligen „getColor"-Funktion in „colors.js" übergeben. Dort sind die Farbrampen hinterlegt, nach denen die Informationen klassifiziert werden.

```
// CO2
L.geoJson(COUNTRY).addTo(overlays.coTwo)
overlays.coTwo.addTo(map)

function styleCo(feature) {
    return {
        fillColor: getColorCo(getData(feature.properties.name_long, "co2")),
        weight: 2,
        opacity: 0.5,
        color: 'white',
        dashArray: '3',
        fillOpacity: 0.5
    };
}

geojson = L.geoJson(COUNTRY, {
    style: styleCo,
    onEachFeature: onEachFeature
}).addTo(overlays.coTwo);
```

```
function getColorCo(data) {
        return data > 9999998 ? '#bdbdbd' :
        data > 10000 ? '#b10026' :
        data > 5000 ? '#e31a1c' :
        data > 1000 ? '#fc4e2a' :
        data > 500 ? '#fd8d3c' :
        data > 100 ? '#feb24c' :
        data > 50 ? '#fed976' :
        data > 10 ? '#ffeda0' :
        data > 0 ? '#ffffcc' :
        '#bdbdbd';
}
```

Innerhalb des Screenshots ist auch der Call einer Funktion namens „onEachFeature" zu erkennen. Diese legt Interaktionen fest, die mit den Polygonen vorgenommen werden können. An dieser Stelle wird definiert, dass die Funktion „highlightFeature" mit dem jeweiligen Polygon ausgeführt werden soll, wenn die Maus darüberfährt und „resetHighlight" wenn die Maus den Bereich des Polygons wieder verlässt. Die Funktion „highlightFeature" bewirkt, dass Informationen über das jeweils betreffende Polygon an „info.update" weitergegeben wird. „resetHighlight" nimmt diese entsprechend zurück.

```
function highlightFeature(e) {
    var layer = e.target;
    info.update(layer.feature.properties);
}

function resetHighlight(e) {
    info.update();
}

function onEachFeature(feature, layer) {
    layer.on({
        mouseover: highlightFeature,
        mouseout: resetHighlight,
    });
}
```

Innerhalb der Info – einem Control-Element in Leaflet – werden die Emissionswerte des ausgewählten Landes dargestellt. Die Beschriftung der Information erfolgt zum einen über den „key" „name", wodurch der Name oben angezeigt wird. Die Anzeige der Daten erfolgt über die Funktion „whichLayer", die eine if-Abfrage enthält. Sie nutzt die Leaflet-Methode „hasLayer", um abzufragen welches Overlay geöffnet ist. Ergibt der „Boolean" für einen der Overlays „true" returnt dies das Ergebnis der Funktion „getDataPrint", welche die Parameter „polyName", „dataType" und „einheit" verwertet. In einer weiteren if-Abfrage wird dort ermittelt, ob Daten vorliegen oder nicht. „Keine Daten vorhanden" wird angezeigt, wenn die Zahl 9999999 beträgt - der Wert des Landes „noMatch" - welchem händisch die Länder zugeordnet wurden, für die keine Werte vorlagen. Ist der Wert ein anderer wird die Funktion „getData" gecallt, das Ergebnis auf eine Stelle gerundet und durch die Einheit und Beschreibung, die als Parameter mit gepasst wurde, ergänzt.

```
// Funktion um noMatch zu benennen
function getDataPrint(polyName, dataType, einheit) {
    if (getData(polyName, dataType) === 9999999) {
        return "Keine Daten vorhanden";
    } else {
        return getData(polyName, dataType).toFixed(1) + einheit;
    }
}

// Funktion um zu ermitteln welcher Layer geöffnet ist und entsprechendes in Info anzuzeigen
function whichLayer(props) {
    if (map.hasLayer(overlays.coTwo) === true) {
        return getDataPrint(props.name_long, "co2", " Mio. t </b><br/> CO<sub>2</sub>-Emission pro Jahr");
    } else if (map.hasLayer(overlays.coTwoGlobalShare) === true) {
        return getDataPrint(props.name_long, "share_global_co2", " % </b><br/> der globalen Emissionen
        pro Jahr");
    } else if (map.hasLayer(overlays.coTwoPerCapita) === true) {
        return getDataPrint(props.name_long, "co2_per_capita", " t </b><br/>  CO<sub>2</sub>-Emissionen
        pro Person und Jahr");
    } else if (map.hasLayer(overlays.coTwoCumu) === true) {
        return getDataPrint(props.name_long, "cumulative_co2", " Mio. t </b><br/> kumulierte Emissionen");
    } else if (map.hasLayer(overlays.coTwoCumuShare) === true) {
        return getDataPrint(props.name_long, "share_global_cumulative_co2", " % </b><br/> der globalen
        kumulierten Emissionen");
    } else {
        return "Keine Daten vorhanden"
    }
}

// Anzeige oben Rechts. Style siehe CSS
var info = L.control();
info.onAdd = function (map) {
    this._div = L.DomUtil.create('div', 'info');
    this.update();
    return this._div;
};
info.update = function (props) {
    this._div.innerHTML = (props ?
        '<b>' + props.name + '</b><hr></hr>' +
        whichLayer(props) :
        'Hover über einen Staat');
};
info.addTo(map);
```

In der Legende werden die Klassengrenzen neben einem Kasten mit der jeweiligen Farbe angezeigt. Dafür werden die Klassen eines jeden Overlays in der Funktion „getClasses" definiert. Auch hier wird sich die Leaflet-Methode „hasLayer" zu Nutzen gemacht. Je nachdem welcher Overlay ausgewählt ist werden unterschiedliche Klassengrenzen in die Legende returnt und in einer „Loop" durchlaufen. Auch anhand der Klassengrenzen kann auf die unterschiedlichen Farbrampen zugegriffen werden. Um zu bestimmen welche ausgewählt werden soll, ermittelt die Funktion „getColorName" ebenfalls über

„hasLayer“ die Farbrampe, auf die sich bezogen werden muss. Diese Legende wird der Karte hinzugefügt.

```
// Legende

// Klassengrenzen
function getClasses() {
    if (map.hasLayer(overlays.coTwo) === true) {
        classes = [0, 10, 50, 100, 500, 1000, 5000, 10000];
        return classes;
    } else if (map.hasLayer(overlays.coTwoGlobalShare) === true) {
        classes = [0, 0.5, 1, 5, 10, 15, 20, 25];
        return classes;
    } else if (map.hasLayer(overlays.coTwoPerCapita) === true) {
        classes = [0, 1.5, 5, 10, 15, 20];
        return classes;
    } else if (map.hasLayer(overlays.coTwoCumu) === true) {
        classes = [0, 500, 1000, 5000, 10000, 50000, 100000, 200000, 400000];
        return classes;
    } else if (map.hasLayer(overlays.coTwoCumuShare) === true) {
        classes = [0, 0.5, 1, 5, 10, 15, 20];
        return classes;
    } else {
        return [0]
    }
}

// Zugriff auf verschiedene Farbrampen
function getColorName(nr) {
    if (map.hasLayer(overlays.coTwo) === true) {
        return getColorCo(nr);
    } else if (map.hasLayer(overlays.coTwoGlobalShare) === true) {
        return getColorCoGlobalShare(nr);
    } else if (map.hasLayer(overlays.coTwoPerCapita) === true) {
        return getColorCoPerCapita(nr);
    } else if (map.hasLayer(overlays.coTwoCumu) === true) {
        return getColorCoCumu(nr);
    } else if (map.hasLayer(overlays.coTwoCumuShare) === true) {
        return getColorCoCumuShare(nr);
    } else {
        return 'not found'
    }
}

// Legende hinzugefuegt
var legend = L.control({
    position: 'bottomright'
});

legend.onAdd = function (map) {
    var div = L.DomUtil.create('div', 'info legend'),
        classes = getClasses(),
        labels = [];
    for (var i = 0; i < classes.length; i++) {
        div.innerHTML +=
            '<i style="background:' + getColorName(classes[i] + 1) + '"></i> ' +
            classes[i] + (classes[i + 1] ? ' - ' + classes[i + 1] + '<br>' : ' +');
    }
    return div;
};

legend.addTo(map);
```

Bei jedem Wechsel des Overlays soll die jeweils passende Legende angezeigt werden. Um dies zu erreichen wurde eine Funktion erstellt, die die bestehend Legende löscht und eine neue einfügt, wann immer das Overlay gewechselt wird. Die Legende kann dabei immer als „legend“ bezeichnet werden und es sind keine separaten Legenden zu entwerfen, da sich diese durch die oben beschriebenen Funktionen selbst anpasst.

```
// Funktion, dass sich bei jedem Layer-Switch die Legende ändert
map.on('baselayerchange', function (eventLayer) {
    if (eventLayer.name === 'CO<sub>2</sub>-Emission pro Jahr') {
        this.removeControl(legend);
        legend.addTo(this);
    } else if (eventLayer.name === 'Anteil jährlicher Emissionen') {
        this.removeControl(legend);
        legend.addTo(this);
    } else if (eventLayer.name === 'Emissionen pro Person') {
        this.removeControl(legend);
        legend.addTo(this);
    } else if (eventLayer.name === 'Kumulierte Emissionen') {
        this.removeControl(legend);
        legend.addTo(this);
    } else if (eventLayer.name === 'Anteil kumulierter Emissionen') {
        this.removeControl(legend);
        legend.addTo(this);
    } else {
        this.removeControl(legend);
        legend.addTo(this);
    }
});
```

main.css

Für ein einheitliches Design erfolgt das Vorgehen für das Styling und die Formatierung der index.html Seite wie bei der style.css der Startseite. Einige weitere Elemente klären den Style der Karte und seiner Komponenten, wie der Info und der Legende. Da das Vorgehen hier bereits für die Hauptseite ausführlich erklärt wurde, wird an dieser Stelle nicht genauer darauf eingegangen.

SDG Watch Austria – Mitgliedsorganisationen

Für die SDG Watch Austria Karte wurde ein Ordner namens „bestpractice“ erstellt, der alle verwendeten Dateien beinhaltet.

Auf der SDG Watch Austria Karten sollen best practice Beispiele abgebildet werden, die aufzeigen, wie und wo gegen die Klimakrise vorgegangen, und Nachhaltigkeit praktiziert wird. Der Grundgedanke war einige weltweit verteilte Beispiele aufzuzeigen. Hier ist jedoch recht schnell deutlich geworden, dass allein eine Auswahl, um passende Projekte zu wählen, viel Zeit in Anspruch nehmen würde. Ebenso wäre es hier schwierig geworden, die Auswahl der Projekte zu begründen.

Der Fokus der dargestellten Projekte wurde somit auf Österreich gerichtet. So wurde die Website der SDG Watch Austria herangezogen. Diese besteht aus zivilgesellschaftlichen und gemeinnützigen Organisationen in Österreich, die sich die Umsetzung der 17 Sustainable Development Goals der United Nations, sowie die Agenda 2030, mit einem Österreichbezug zum Ziel gesetzt haben (SDG Watch Austria).

Über 200 Organisationen sind österreichweit Mitglied der SDG Watch. Diese sind unterteilt in verschiedene Tätigkeitsbereiche. Die Website ermöglicht das Filtern der Organisationen nach diesen Kategorien. Es steht jedoch keine Datei mit den einzelnenn Organisationen und zugehörigen Informationen zur Verfügung, weshalb die Datenbeschaffung für unsere Karte händisch ablaufen muss. Es würde den zeitlichen Rahmen dieser Projektarbeit sprengen, alle Organisationen mit in die Karte aufzufassen. So wurden die vier Kategorien: Stadtentwicklung, Gesundheit, Soziales und Umwelt, Klima, Landwirtschaft & Tierschutz ausgewählt. Gemeinsam beinhalten sie 109 Organisationen, jedoch sind wenige auch in mehreren Kategorien vertreten und doppeln sich somit.

Datenbeschaffung:

Um diese Organisationen für unser Projekt brauchbar in Daten umzuwandeln, wurde im Projekt eine neue Datei ‚mitglieder.js‘ erstellt. In diese wurden zu allen Organisationen Daten wie Name, Adresse, Mail, Kurzüberschrift, kleine Beschreibung und der Link zur Homepage gesammelt. Die Daten stammen von den jeweiligen Homepages der Organisationen und wurden Großteiles dort zusammengesucht.

Untergliedert in einzelne Konstanten (‚const‘) nach Kategorie wurden die Daten in Arrays mit nested Einträgen in eckigen Klammern zusammengetragen und mit Koordinaten versehen, damit auf alle einzelnen Einträge und deren Parametern nach Kategorie zugreifen kann und sie mit den Koordinaten verortet werden können. Für die Koordinaten wurde die Adresse, die auf der Homepage der Organisationen angegeben war, in Google Maps eingegeben und die lat/long Werte herangezogen. Dies ist beispielhaft in der folgenden Abbildung dargestellt.

```
const SOZIALES = [
    {
        nr: 1,
        user: "Concordia Sozialprojekte",
        Adresse: "Hochstettergasse 6, 1020 Wien",
        Mail: "office@concordia.or.at",
        Öffnungszeiten: "",
        intro: "Gemeinnützige Privatstiftung",
        about: "CONCORDIA ist eine internationale, un
        weblink: "https://www.concordia.or.at/",
        lat:  48.22516071255066,
        lng: 16.388036007608115,
    },
```

index.html

Die index.html Datei wurde in starker Anlehnung an die der Hauptseite angepasst. So kann eine einheitliche Darstellung gewährleistet werden. Für Plugins wurden dementsprechende Referenzen eingefügt. Dazu mehr im Teil ‚main.js', in welchem die Plugins in ihrer Funktion und Anwendung erläutert werden.

Nach einem Headerbild, das eine Blumenwiese zeigt, wurde eine Navigationsleiste hinzugefügt, um zur Startseite oder zur ersten Karte zu gelangen. Hierfür wurde die genaue Referenz der Zielseite mit einem Style eingebunden. Für die Einheitlichkeit wurde der Bescheribungstext der Karte mit den definierten Klassen (‚class') der anderen beiden Seiten gestyled. In einem footer wurde erneut durch das Einbinden von Referenzen und Styleangaben eine Funktion errichtet, die es ermöglich zu Karte 1 oder zur Startseite zu gelangen.

main.css

Die main.css Datei wurde für eine Einheitlichkeit hauptsächlich von der style.css Datei übernommen. Überschriften wurden hier jedoch für die Texte in den Popups neu erstellt. Damit die Position der Layer in der Layer control Box linksbündig ist, diese jedoch nicht spezifisch angesteuert werden kann, wurde text-align für den gesamten main Teil auf linksbündig gestellt. Anschließend wurde, die Bausteine, die für die restlichen Texte zuständig sind, wieder zentriert. In diesem Zug wurden auch Schrift- und Hintergrundfarben der Popups umgeschrieben und ansprechend gestaltet.

```
30  main {
31      text-align: left;
32      margin: 16px;
33  }
```

```
64  main h3 {
65      background-color: #c1f1f3;
66      color: black;
67      text-align: center;
68      font-weight: bold;
69      font-size: 17px;
70  }
```

main.js

Die main.js Datei wurde damit begonnen, dass Bounds erstellt wurden. Diese Bounds bestimmen, bis zu welchem Ausmaß die Baselayer Karte bewegt werden kann. Hier wurden Koordinaten gewählt, die eng genug sind, dass nicht ganz Europa abgebildet werden kann, aber auch nicht zu eng sind, da sie sonst mit der Popup-Funktion kollidieren.

```
3  //create bounds, so map keeps focus on Austria
4  let bounds = [
5      [43, 8], // Southwest coordinates
6      [51.2, 18] // Northeast coordinates
7  ]
```

Darauffolgend wurde die Karte als Konstante ‚map' erstellt. Der erste Plugin ist hier die fullscreenControl. Referenzen dazu wurden im index.html verknüpfend eingefügt. Mit diesem Plugin wird ein Symbol in der Karte kreiert, über welches die Karte im Vollbildmodus betrachtet werden kann. Über ‚maxBounds' wurden die zuvor definierten Bounds als maximale Ausdehnung des Kartenausschnittes gewählt. Mit zoomControl: true wurden + und – zum Zoomen der Karte hinzugefügt. Durch setView nach den dargestellten Optionen wird der Ausschnitt definiert, welcher beim erstmaligen Öffnen der Seite erscheint. Hierfür wurden die Koordinaten 47.791 und 13.217 mit einer Zoom Level von 7 ausgewählt. Diese Werte basieren auf vielen Versuchen, den perfekten Ausschnitt zu definieren.

Mit L.tileLayer wird eine über Leaflet zur Verfügung stehende Openstreetmap als Baselayer der Karte hinzugefügt. Unter attribution werden die Quellen der Karte, der Icons und die SDG Watch Austria verlinkt und permanent im unteren rechten Eck der Karte angezeigt.

Folglich wurde noch ein Zoom Stopp eingefügt, der verhindert, dass weiter als die gesetzten Bounds aus der Karte raus gezoomed werden kann. Dies war zuvor trotz Bounds noch möglich.

```
//create map with options
const map = L.map('map', {
    fullscreenControl: true, //option to viep map on fullscreen
    maxBounds: bounds, //set bounds as maximum extend
    zoomControl: true //option to zoom in and out on map
}).setView([47.791, 13.217], 7);


L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>, <a href="https://mapicons.mapsm
}).addTo(map);


//min Zoom: stops zooming out at one point
map.setMinZoom(map.getBoundsZoom(map.options.maxBounds));
```

Für die darzustellenden Kategorien der Mitglieder der SDG-Watch Austria werden Overlays erzeugt, die an und ausgeschaltet werden können. Mit dem Plugin markerClusterGroup werden einzelne Einträge pro Kategorie je nach Zoom zu Clustern zusammengefasst dargestellt. Die dazugehörige Leaflet Referenz wurde im index.html verankert.

```
//create overlays for categories with function: on/off
let overlays = {
    UmweltKlima: L.markerClusterGroup(),
    Stadtentwicklung: L.markerClusterGroup(),
    Gesundheit: L.markerClusterGroup(),
    Soziales: L.markerClusterGroup(),
};
```

Im nächsten Schritt wurden mit der Layer Control die Overlays den vorhandenen Layern hinzugefügt. Hier konnte im selben Zug wie die Namensgebung der Layer auch ein bild eingefügt werden. So wurden die Icons, die später auf der Karte die Orte markieren, wie bei einer Legende hinzugefügt werden. Die LayerControl wurde in das rechte obere Eck der Karte gesetzt (position: ‚topright') und mit der Maus ausklappbar dargestellt (collapsed: true). Diese Layer wurden folglich der Karte hinzugefügt. In einem weiteren Schritt wurden die Overlays der Karte hinzugefügt (.addTo(map)).

```
//  add overlays to layers
let layerControl = L.control.layers({}, {
    "<img src='icons/city.png' /> Stadtentwicklung": overlays.Stadtentwicklung,
    "<img src='icons/health.png' /> Gesundheit": overlays.Gesundheit,
    "<img src='icons/sozial.png' /> Soziales": overlays.Soziales,
    "<img src='icons/tree.png' /> Umwelt, Klima, Landwirtschaft & Tierschutz": overlays.UmweltKlima
}, {
    position: 'topright',
    collapsed: true,
}).addTo(map);
```

Um die Positionen der Mitglieder auf der Karte ansprechend darstellen zu können, wurden Icons erstellt. Diese wurden über https://mapicons.mapsmarker.com/ kostenlos und zu freien Verfügung stehend heruntergeladen und im selben Ordner wie die restlichen Dateien auch gespeichert (Map Icons Collection). Die Logos wurden nach Symbolen passend zu den vier Kategorien ausgesucht und farblich leicht zu unterscheiden dargestellt. Bei zukünftigen Karten könnte zusätzlich bei der Farbwahl eine rot-grün-Sehschwäche berücksichtigt werden, um die Karte etwas barrierefreier zu gestalten.

So wurde die Variable LeafIcon erstellt und mit den Optionen unter L.Icon.extend wurde die Größe der Icons, der Ankerpunkt der Icons und der Ankerpunkt der dazugehörigen Popups festgelegt. Folgend wurde pro Kategorie eine neue Variable mit passendem Name erstellt, welche die Optionen von LeafIcon aufweist, jedoch ein eigenes, passendes Icon zugewiesen bekommt.

```
//create icons
var LeafIcon = L.Icon.extend({
    options: {
        iconSize: [38, 38], // size of the icon
        iconAnchor: [16, 37], // point of the icon which will correspond to marker's location
        popupAnchor: [2, -38] // point from which the popup should open relative to the iconAnchor
    }
});

var UmweltIcon = new LeafIcon({
        iconUrl: 'icons/tree.png'
    }),
    SozialIcon = new LeafIcon({
        iconUrl: 'icons/sozial.png'
    }),
    StadtIcon = new LeafIcon({
        iconUrl: 'icons/city.png'
    });
GesundheitIcon = new LeafIcon({
    iconUrl: 'icons/health.png'
});
```

Nun steht das Grundgerüst so weit, dass die Daten aus den Arrays der Kategorien eingelesen und den Overlays und Layern zugeteilt werden können. Hierfür wird eine for Schleife geschrieben, die auf die einzelnen Einträge der in mitglieder.js definierten Konstanten Kategorien zugreift. In der Schleife wird zuerst ein Marker des Eintrages definiert. Dieser wird über die Abfrage nach den gespeicherten lat/long Daten auf der Karte verortet und ihm wird das passende Icon zugewiesen. In einem bindPopup das an den Marker gebunden wird, werden die Einträge hinzugefügt, die in der Karte bei einem Klick auf ein Icon angezeigt werden sollen, definiert. Hier wird mit String Interpolationen gearbeitet. Um die Mailadresse mit einem Popup-Mailprogramm zu verbinden, wird sie als href angegeben und ein kleiner Briefumschlag als Icon wird davor hinzugefügt. Dasselbe wird mit dem Weblink der Organisation gemacht. Hier hat es jedoch nicht funktioniert, dass bei einem Klick ein neuer Tab geöffnet wird, etwas hat mit target:"_blank" nicht gepasst. Jedoch funktioniert es, dass sich im selben Tab die Website der Organisation öffnet. Mit der Einstellung maxHeight wird die maximale Höhe des Popup Fensters angegeben, je nach Größe wird der Text dann automatisch zum Scrollen in der Box angezeigt. Hier wurde eine Höhe von 320 gewählt, da größere Popups mit den gesetzten Bounds der Karte kollidierten. Das Gesamte wird schlussendlich dem jeweiligen Kategorie Overlay zugewiesen.

```
//tear data from mitglieder.js and add to map with marker and popup
for (let entry of UMWELT) {
    let mrk = L.marker([entry.lat, entry.lng], {
        icon: UmweltIcon
    });
    mrk.bindPopup(`<h1>${entry.user}<h1>
        <h3>${entry.intro}</h3>
        <h4>${entry.about}</h4>
        <h4>Adresse: ${entry.Adresse}</h4>
        <h4><i class="far fa-envelope mr-3" style="margin-right: 0.3em"></i><a href=" mailto:${entry.Mail}" t
        <p><a href="${entry.weblink}" target:"_blank"><i class="fas fa-external-link-alt mr-3" style="margin-
        `, {
        maxHeight: 310
    }).addTo(overlays.UmweltKlima);
}

for (let entry of SOZIALES) {
    let mrk = L.marker([entry.lat, entry.lng], {
        icon: SozialIcon
```

Folgend wurde ein weiteres Plugin eingebaut, L.Hash. Dies sorgt dafür, dass die URL der Karte je nach Position, auf welchen der Zoom liegt, dynamisch angepasst wird. Die unterstützt das Lenken von Usern

auf spezifische Kartenausschnitte, die im URL angegeben sind (Leaflet-hash). Auch hierfür war es notwendig, ein Script Tag von cdnjs.com in die index.html Datei einzubinden.

```
139    // Leaflet hash
140    var hash = new L.Hash(map);
```

Zur besseren Orientierung auf der Karte wurde eine Minimap in die rechte untere Ecke hinzugefügt. Diese ist einklappbar und zeigt nur Österreich an, da für diese Karte nur Österreich von Relevanz ist und somit ein Überblick noch besser gewährt werden kann. Hierfür war ebenso das Einbinden eines Script Tags con cdnjs.om in die index.html Datei notwendig.

Zu guter Letzt wurde ein dynamischer Maßstab der Karte hinzugefügt, da dies ein relevantes Objekt einer Karte ist, bei welcher Positionen und auch Abstände zwischen Objekten eine Rolle spielen kann. Hierfür wurde ein Feature von Leaflet genutzt (L.control.scale). Für diese konnte die Maßeinheit auf metrisch gestellt werden und die Position gewählt werden. Damit der Maßstab nicht einzeln auf der Karte verloren geht, wurde er direkt unter die Zoomleiste gesetzt und der Karte hinzugefügt.

```
150  //Add Scale bar
151  L.control.scale({
152      metric: true,
153      imperial: false,
154      position: ('topleft')
155  }).addTo(map);
```

Fazit & Ausblick

Insgesamt sind wir mit den Webseiten und den Karten sehr zufrieden. Das Layout ist einheitlich und alle geplanten Funktionen funktionieren auf den Karten.
Schwierigkeiten ergaben sich unter anderem bei dem Design der Layer Control anzeige, der Darstellung der Icons und Popups, damit sie trotz Bounds und clustern an der richtigen Stelle angezeigt werden, sowie allgemein bei der Gestaltung der Responsivität aller Elemente auf verschiedenen Gerätegrößen.

Aus zeitlichen Gründen haben wir die dritte geplante Karte (über die Klimafolgen) nicht umgesetzt, um uns auf die Qualität der Webseiten und der anderen zwei Karten zu fokussieren.

Bei der Länderspezifische Emissions-Daten Karte hat sich vor allem die Manipulation der Interaktions-Funktionen wie „highlightFeature" oder „style" als problematisch herausgestellt. Anders als bei anderen Funktionen war hier das Passen zusätzlicher Parameter nicht möglich (bzw. entsprechende Versuche haben nicht funktioniert), wodurch es nötig war den Style der Overlays einzeln zu definieren. Hier könnte Code gespart werden.

Innerhalb der Funktionen hat sich „resetHighlight" zudem als fehleranfällig herausgestellt. Eigentlich war geplant, dass sich der Style eines Polygons beim Herüberfahren verändert. Aufgrund der oben beschriebenen Fülle an Styles gab es jedoch das Problem, dass der Style beim Entfernen der Maus zu einem anderen Style zurückgesetzt wurde. Falls sich das oben beschrieben Problem lösen ließe, könnte hier mit „map.hasLayer" eventuell der Effekt wieder eingefügt werden.
Innerhalb des Datensatzes gab es teilweise Unstimmigkeiten, weswegen die Erstellung des „Landes" „noMatch" notwendig war. Mit mehr Zeit wären hier eventuell elegantere Lösungen möglich gewesen. Zudem könnten die Namen der Länder ins Deutsche übersetzt werden, um die Einheitlichkeit der Seite zu wahren. Entsprechende Schritte könnten in einer weiteren Entwicklungsphase durchgeführt werden.

Für die SDG Watch Austria Karte gibt es noch einige Plugins, Funktionen und Elemente, die mit mehr Zeit und Know-how noch in die Karte eingebaut werden könnten, um sie noch attraktiver zu machen. Eine Suchfunktion, über welche nach einzelnen Mitgliedsorganisationen gesucht werden kann wurde versucht zu installieren, jedoch gab es Probleme mit der Leaflet Suchfunktion. Alternativ könnte ein Pulldow-Menü eingebaut werden, jedoch könnte das mit so vielen Einträgen auch schnell unübersichtlich sein. Ebenso könnte eine Bookmark Funktion eingebaut werden, über welche Organisationen gespeichert werden können.
Was bisher leider noch nicht funktioniert hat ist, dass sich bei dem Klick auf den URL der Organisation im Popup ein neuer Tab öffnet. Mit target: "_blank" hat dies leider nicht funktioniert, vielleicht muss es anders eingebaut werden, da die Einträge direkt davor über ${entry.} abgerufen wurden und nicht einfach ein URL dasteht.
Mit mehr Zeit könnten alle +200 Mitgliedsorganisationen der SDG Watch Austria in die Arrays eingespeist werden, damit die Karte eine gewisse Vollständigkeit aufweist.

Allgemein ist uns aufgefallen, dass das Projekt nie fertig scheint, weil uns immer neue Ideen und Verbesserungsmöglichkeiten einfallen. Wir sind jedoch mit unserem Ergebnis sehr zufrieden und freuen uns, dass wir in einem Semester so viel lernen konnten, dass wir dieses Projekt selbstständig erstellen konnten.

Quellen

Ash Kyd: https://geojson-maps.ash.ms/ [letzter Zugriff: 16.06.2021].

Font Awesome: https://fontawesome.com/v5.15/icons?d=gallery&p=2 [letzter Zugriff: 16.06.2021].

Google Font Raleway: https://fonts.google.com/specimen/Raleway [letzter Zugriff: 16.06.2021].

Leaflet: https://leafletjs.com/download.html [letzter Zugriff: 16.06.2021].

Leaflet Fullscreen: https://github.com/Leaflet/Leaflet.fullscreen [letzter Zugriff: 16.06.2021].

Leaflet Hash: https://github.com/mlevans/leaflet-hash?utm_source=cdnjs&utm_medium=cdnjs_link&utm_campaign=cdnjs_library [letzter Zugriff: 16.06.2021].

Leaflet MarkerCluster: https://github.com/Leaflet/Leaflet.markercluster [letzter Zugriff: 16.06.2021].

Leaflet MiniMap: https://github.com/Norkart/Leaflet-MiniMap [letzter Zugriff: 16.06.2021].

Leaflet Providers: https://github.com/leaflet-extras/leaflet-providers [letzter Zugriff: 16.06.2021].

Leaflet-Tutorial - Interactive Choropleth Map: https://leafletjs.com/examples/choropleth/ [letzter Zugriff: 16.06.2021].

Map Icons Collection: https://mapicons.mapsmarker.com/ [letzter Zugriff: 16.06.2021].

Our World in Data (OWID): https://github.com/owid/co2-data [letzter Zugriff: 16.06.2021].

SDG Watch Austria: https://www.sdgwatch.at/de/wer-wir-sind/sdg-watch-austria/ [letzter Zugriff: 15.06.2021].