

**Kartografie II FS 2021****Übung 6: Scripting mit JavaScript**

29.04.2021

**Übungsverantwortung**

---

Magnus Heitzler  
Michael Schmuki[hmagnus@ethz.ch](mailto:hmagnus@ethz.ch)  
[mschmuki@ethz.ch](mailto:mschmuki@ethz.ch)HIL G 13.2  
HIL G 12.1**Übungsziel**

---

Das Ziel dieser Übung ist die Erweiterung der in Übung 5 erstellten Webkarte um ein Histogramm, welches die Anzahl der Länder nach Klasse wiedergibt. Hierzu werden einfache Funktionen mithilfe von JavaScript geschrieben und in das HTML-Dokument eingebunden. Dabei soll insbesondere der Umgang mit dem Document Object Model (DOM) sowie die Anpassung von Attributen erlernt werden.

Abgegeben werden soll die erstellte Webkarte (HTML-Dokument (.html), CSS-Dokument (.css)) mitsamt der erstellten JavaScript-Datei (.js).

**Dokumentationen**

---

Dokumentationen:

[HTML](#)  
[CSS](#)  
[JavaScript](#)

Cheatsheets:

[HTML](#)  
[CSS](#)  
[JavaScript](#)

Mozilla Development Network:

[MDN \(en\)](#)  
[MDN \(de\)](#)**Ausgangslage und Grundlagedaten**

---

Die zu erstellende interaktive Webkarte soll auf den Resultaten von Übung 5 aufbauen. Diese sind:


- Die exportierten Polygone der Länder
- Die Webkarte (map.html, map.css)

Für diejenigen, die die Webkarte von Übung 5 nicht fertiggestellt haben, kann eine Musterlösung von Moodle heruntergeladen werden.

### Aufgabe A1: Das Document Object Model (DOM)

In dieser Aufgabe geht es zunächst darum, sich mit der Manipulation des DOMs mithilfe des der Webdeveloper-Tools des Browsers vertraut zu machen.

#### 1.1 DOM-Zugriff mit dem Browser

- HTML-Dateien sind, wie viele andere für das Web relevante Dateiformate (z.B. XML), hierarchisch aufgebaut. Der Zugriff auf diese Struktur erfolgt über das sogenannte Document Object Model (DOM). Aufgrund dieser hierarchischen Struktur wird auch häufig von dem sogenannten DOM-Baum gesprochen.
- Jede Komponente dieses Baumes ist ein Knoten («Node»), der Kindknoten, Geschwisterknoten, etc. besitzen kann. Elemente sind eine spezielle Art von Knoten, die die eigentliche Struktur des HTML-Dokumentes ausmachen und auf die in der Regel über ihre Id oder Klasse zugegriffen wird. Knoten, die keine Elemente sind, sind beispielsweise Kommentare und Text.
- Öffne nun die von dir in Übung 5 erstellte Webkarte in einem Browser. Nutze alternativ die bereitgestellte Musterlösung.
- Während das DOM grafisch im Browser-Fenster dargestellt wird, kann auch auf die interne Struktur direkt im Browser zugegriffen werden. Dies erfolgt über die sogenannten Webdeveloper-Tools, die in Firefox, Chrome und Edge mit dem Shortcut [F12] aufgerufen werden können. Wenn du einen anderen Browser verwendest, musst du ggf. kurz recherchieren, wie du die Webdeveloper-Tools öffnen kannst.
- Klicke in den Webdeveloper-Tools auf den Reiter «Elements», um das DOM in Text-Form anzuzeigen. Beachte, dass die einzelnen Elemente auf- und zugeklappt werden können. Zudem kann hier direkt eingesehen werden, welche Style-Anweisungen auf einzelne Elemente wirken. Klicke hierzu auf eines der aufgelisteten Elemente. Zudem kann hier interaktiv das DOM manipuliert werden (z.B. über Doppelklick in den Titel-Text). Weitere Manipulationsmöglichkeiten werden über einen Rechtsklick angezeigt.
- Ein weiteres nützliches Werkzeug, um das DOM zu untersuchen, ist der «Inspector», welcher eine grafische Selektion einzelner Elemente im Browser-Fenster zulässt. Er kann über ein Icon aktiviert werden, das in etwa so aussieht: .
- Mit dem Inspector selektierte Elemente werden direkt im Reiter «Elemente» hervorgehoben.
- Mach dich mit der Funktionsweise dieser Art der DOM-Manipulation vertraut.

#### 1.2 DOM-Zugriff mit JavaScript

- Im Fall von interaktiven Webseiten wird das DOM mit JavaScript verändert. Praktischerweise kann JavaScript direkt im Browser ausgeführt werden, sodass Änderungen im DOM direkt sichtbar werden, ohne dass die Webseite neugeladen werden muss.
- Öffne zunächst die «Konsole» in den Webdeveloper-Tools.
- Tippe zunächst schlichtweg `document` in die Konsole ein. Ausgegeben werden Informationen zu dem JavaScript-Objekt, welches einen Zugriff auf das HTML-Dokument ermöglicht. Es stellt sozusagen den «Stamm» des DOM-Baumes dar, von dem aus auf alle anderen Objekte zugegriffen werden kann.
- Beispielsweise kann über `document.head` auf den HTML-Header und über `document.body` auf den eigentlichen Inhalt zugegriffen werden. Diese stellen zwei vom «Stamm» abzweigenden «Äste» dar, wobei insbesondere der `body`-«Ast» viele weitere abzweigende Äste (z.B. Div-Elemente) besitzen kann, die jeweils wiederum Äste besitzen können.
- Es sind mit JavaScript allerdings auch komplexere Abfragen möglich. So kann beispielsweise über den Befehl `let my_title = document.getElementById("title")` auf ein Element mit der Id `title`, in diesem Fall eben auf den Div-Container der die Überschrift enthält, zugegriffen werden. Passe diese Id ggf. an, wenn dein Überschrift-Element eine andere Id besitzt. Das zurückgegebene Objekt wird in der Variable `my_title` gespeichert. Informationen über dieses Objekt können wiederum mithilfe von `console.dir(my_title)` ausgegeben werden. Über `my_title.innerHTML = "Alternativer Titel"` kann beispielsweise der Titel geändert werden. Probiere diese Befehle am besten einfach aus. Lege allenfalls zunächst eine Id für ein Objekt an.
- Objekte können auch anhand ihrer Klasse selektiert werden. Hierbei kann in der Musterlösung der folgende Ausdruck verwendet werden:

```
let legend_items = document.getElementsByClassName("legend_item");
```

Die Variable `legend_items` enthält in diesem Fall eine Liste der Div-Container, die die einzelnen Einträge der Legende ausmachen, da sie alle dieselbe Klasse besitzen. Der Zugriff auf eine Liste erfolgt üblicherweise über Indices. Beispielsweise würde der Ausdruck `legend_items[0]` das erste Element der Liste zurückgeben, `legend_items[1]` das zweite Element und so weiter. Der schrittweise Zugriff über sämtliche Elemente erfolgt üblicherweise mithilfe einer Schleife. Diese könnte beispielsweise folgendermassen aussehen:

```
for (let i = 0; i < legend_items.length; i++)
{
    console.dir(legend_items[i]);
}
```

In diesem Fall werden nacheinander die Informationen zu jedem Element, in diesem Fall der Div-Container der Legende, in der Konsole ausgegeben.

- Daneben ist es auch möglich auf Kindelemente, Geschwisterlemente, etc. zuzugreifen. Folgendes Script gibt beispielsweise die Kind-Elemente (`children`) sowie sämtliche Kind-Knoten (`childNodes`) des Div-Containers der Legende aus:

```
let legend_container = document.getElementById("legend_container");
console.dir(legend_container.children);
console.dir(legend_container.childNodes);
```

## Aufgabe A2: Erstellung einer JavaScript-Datei

In dieser Aufgabe liegt der Fokus auf der Erstellung eines Scriptes, um die bestehende Webkarte dynamisch um ein Balkendiagramm zu ergänzen.

### 1.1 Einbinden einer JavaScript-Datei

- Erstelle eine leere Textdatei mit dem Namen `map.js` in demselben Ordner, in welchem sich deine Datei `map.html` befindet. Diese Datei soll den Code beinhalten, um das Histogramm zu erzeugen.
- Füge zunächst die folgende Zeile in `map.js` ein:

```
console.log("It works!");
```

- Binde nun die Datei `map.js` in das Dokument `map.html` ein, indem du folgende Zeile in den Header einfügst:

```
<script src="map.js"></script>
```

- Lade nun die Datei `map.html` in einem Browser und öffne die Webdeveloper-Tools. Wechsle dort in die «Konsole».
- Drücke den «Refresh»-Knopf (oder Shortcut `[CTRL]+[R]`), sodass die Karte bei offener Konsole neu geladen wird. In der Konsole sollte nun der Text «It works!» erscheinen.

### 1.2 Erstellen der `onload`-Funktion

- Als nächstes soll eine Funktion erstellt werden, die aufgerufen wird, sobald das gesamte Dokument vollständig geladen wurde. In dieser Funktion wird die wesentliche Logik zur Erstellung des Histogramms hinterlegt.
- Füge zu diesem Zweck folgende Zeilen **ganz an den Anfang der JavaScript-Datei** ein:

```
window.onload = function() {
    console.log("Loaded!");
};
```

Wenn nun die Karte im Browser neu geladen wird, solltest du feststellen, dass zunächst «It works!» erscheint und anschliessend erst «Loaded!», obwohl «Loaded!» eigentlich im Code vor «It works!» vorkommt. Der Grund hierfür ist, dass der Inhalt der JavaScript-Datei direkt ausgeführt wird, sobald der Browser die Zeile `<script src="map.js"></script>` liest. Die darin enthaltene Funktion `window.onload` wird jedoch erst aufgerufen, sobald die Seite vollständig geladen wurde!

### Aufgabe B1: Datengrundlage

#### 1.1 Anlegen eines JavaScript-Objektes

- Lege in der onload-Funktion von map.js ein neues Objekt an, welches für jeden Klassennamen eine Liste mit zwei Einträgen enthält. Diese Datenstruktur soll später ausgelesen werden bzw. als Grundlage für die Konstruktion eines Balkendiagramms dienen.
  - o Verwende für das erste Element in der Liste eine beliebige, Ganz-Zahl als Platzhalter (Im nachfolgenden Beispiel wurden dazu die Zahlen 10-15 verwendet)
  - o Verwende für das zweite Element in der Liste die Beschreibung der entsprechenden Klasse als String, welche du von der letzten Übung übernehmen kannst.
- Weise dieses Objekt der Variable «histogram» zu
- Ein Beispiel könnte folgendermassen aussehen:

```
let histogram = {
  very_low: [10, "<1000"],
  low: [11, "1000-2500"],
  medium: [12, "2500-10000"],
  high: [13, "10000-25000"],
  very_high: [14, ">25000"],
  no_data: [15, "no data"]
};
```

- Vergewissere dich, dass das Objekt korrekt initialisiert wurde indem du es im Anschluss an die Initialisierung auf der Konsole ausgibst:

```
console.log(histogram);
```

#### 1.2 Zählen der Länder-Klassen mit JavaScript

- Im nächsten Schritt gilt es die Platzhalter-Zahlen (10-15 im obigen Beispiel) zu entfernen und stattdessen die effektive Anzahl an Klassen dynamisch zu ermitteln.
  - o *Hinweis:* Verwende dazu [getElementsByClassName](#) um eine Liste der entsprechend Klassifizierten Nodes im SVG-Element zu erhalten
  - o *Hinweis:* Verwende [Array.length](#) auf der zurückgegebenen Liste um die Anzahl Elemente in der Liste auszulesen.
- Vergewissere dich, dass die Anzahl der Länder pro Klasse korrekt ermittelt wurde. Gebe dazu das Objekt erneut in der Konsole aus:

```
console.log(histogram);
```

- Vergleiche die dynamisch ermittelten Werte in der Konsole stichprobenartig mit der effektiven Anzahl der Länder indem du die Such- bzw. Zähl-Funktion deines Editors im .html-Dokument verwendest (z.B. Suche nach `class="low"`), oder du die Attribut-Tabelle in QGIS einsiehst und [nach der entsprechenden Klasse filterst](#).

### Aufgabe B2: Erstellung des Balkendiagramms

#### 2.1 Erstellung des Histogramm-Containers

- Erstelle nun ein neues Div-Element in map.html und passe seine Grösse, Position und seinen Stil mithilfe von CSS nach deinen Wünschen an. Er soll als Container für die einzelnen zu erstellenden Balken fungieren.
- Wenn du möchtest, kannst du auch deine bereits erstellte Legende als Grundlage benutzen.

#### 2.2 Dynamische Erzeugung von Balken

- Programmiere nun eine Schleife, welche über die einzelnen Einträge des JavaScript-Objektes iteriert. Dies kann beispielsweise auf folgende Art und Weise geschehen:

```
for (let className in histogram) {
  let classCount = histogram[className][0];
  let label = histogram[className][1];
}
```

- Die Variablen `className`, `classCount` und `label` beinhalten hierbei die Werte des JavaScript-Objektes. Benutze ggf. `console.log()`, um dich mit diesem Konstrukt vertraut zu machen.
- Erstelle nun für jeden Schleifendurchlauf ein neues Div-Element, das als ein Balken des Histogramms dienen soll. Mache hierzu seine Grösse (Höhe oder Breite) von der entsprechenden Länderzahl abhängig. Neue Elemente können beispielsweise folgendermassen erzeugt werden:

```
let myDiv = document.createElement("div"); // neues Div-Element
myDiv.style.height = "20px"; // setzt die Höhe fest
```

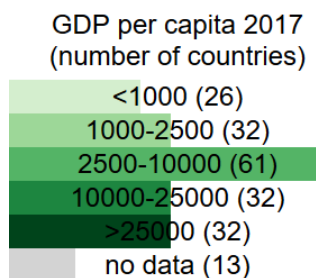
- Neu erstellte Elemente müssen einem Elternelement zugewiesen werden, damit sie im DOM-Baum erscheinen. Dies kann folgendermassen erfolgen:

```
myParentDiv.appendChild(myDiv);
```

- Erzeuge ausserdem eine Beschriftung für jeden Balken. Benutze hierzu beispielsweise den «+»-Operator, um Zahlen und Texte aneinanderzuhängen. Es kann notwendig sein, für die Beschriftung und den Balken separate Div-Elemente zu erstellen und diese in einen Container einzufügen. Text kann beispielsweise einem Div-Container folgendermassen zugewiesen werden:

```
myDiv.innerHTML = "mein Text";
```

- Füge einen Titel hinzu und färbe ggf. die einzelnen Balken entsprechend der Ländersymbolisierung ein.
- Ein Histogramm könnte beispielsweise so aussehen:



## Abgabe

---

### Reminder:

- Die Übung muss bis Mittwoch, 05.05.2021 23:59 Uhr in Moodle hochgeladen werden.

### Abgegeben werden muss eine Zip-Datei mit folgendem Inhalt:

- HTML-Dokument, welches die finale Webkarte beinhaltet, das zugehörige CSS-Dokument sowie das JavaScript-Dokument.

## Beurteilung

---

Die Übung 6 wird gemäss den folgenden Aspekten beurteilt und benotet; dabei tragen die einzelnen Aspekte mit unterschiedlicher Gewichtung zur Gesamtnote bei:

### Beurteilungsaspekt 1: Interaktive Karte (Gesamt 10P)

- JavaScript-Datei wurde erzeugt (0.5P) und in das HTML-Dokument eingebunden (0.5P)
- Die `onload`-Funktion wurde erstellt (1P)
- Das JavaScript-Objekt wurde erstellt und die Anzahl der Klassen wird dynamisch ermittelt (2P)
- Der Histogramm-Container wurde erstellt (0.5P) und wurde an geeigneter Stelle platziert (0.5P)
- Die Balken des Histogramms werden automatisch anhand des JavaScript-Objektes erzeugt und geeignet skaliert (2P)
- Jeder Balken besitzt eine geeignete Beschriftung bestehend aus Klassenbeschriftung und Länderzahl (2P)
- Das Histogramm besitzt einen Titel (1P)