2. javascript

Javascript ist eine ausgewachsene, objektorientierte Programmiersprache. Mittlerweile können Webapplikationen in reinem javascript geschrieben werden.

Konzept
Objektorientierte Programmiersprache kennen Objekte oder Klassen. In unserem Beispiel ist ein Element auf der Webseite ein Objekt. Z. B. das DIV-Objekt. „Divisons“ (div’s) werden benutzt, um die Seite in Abschnitte einzuteilen. Jedes div-Element ist dabei eine Instanz (ein konkretes Erscheinungsbild des abstrakten Abschnitts). Div selbst ist abstrakt, da ein div z.B. einen langen oder kurzen Abschnitt umfasst. Durch Schreiben der Tags und und dadurch, dass wir etwas zwischen Start- und End-Tag schreiben, erzeugen wir eine Instanz des div-Objekts.
Mit javascript können die Eigenschaften von Objekten auf unserer Seite gelesen werden. Objekte haben Methoden („Funktionen“) und Eigenschaften.

Aufgabe:

1. **Erstelle mit einem Texteditor die folgende Datei „kartenprojekt_v1-1.html“ (Änderungen zur vorherigen Datei sind rot):**

```
<!DOCTYPE html>
<html lang="de">
  <head>
     <meta charset="utf-8"/>
    <title>Mein Kartenprojekt</title>

  </head>
  <body>
    <h2>Karte</h2>
    <div id="karte" class="karte">
</div>
     <script>
     var a = document.getElementById("karte");
     console.log(a);
     console.log(typeof(a));
     a.style.width = "200px";
     a.style.height = "200px";
     a.style.background = "green";
     </script>
  </body>
</html>
```

Das Objekt „document“ kann mit der Methode getElementById auf Elemente mit einer bestimmten Identität untersucht werden. Diese Id ist ein EINDEUTIGER Name. Er darf nur einmal auf der Seite vorkommen.
Durch die Anweisung „var a =“ wird das, was die Funktion beim Durchsuchen des Objekts document findet als Inhalt der Variable a gespeichert.
a enthält also nach dieser Anweisung die Instanz des Objekts, das die Id „karte“ hat.

Aufgabe

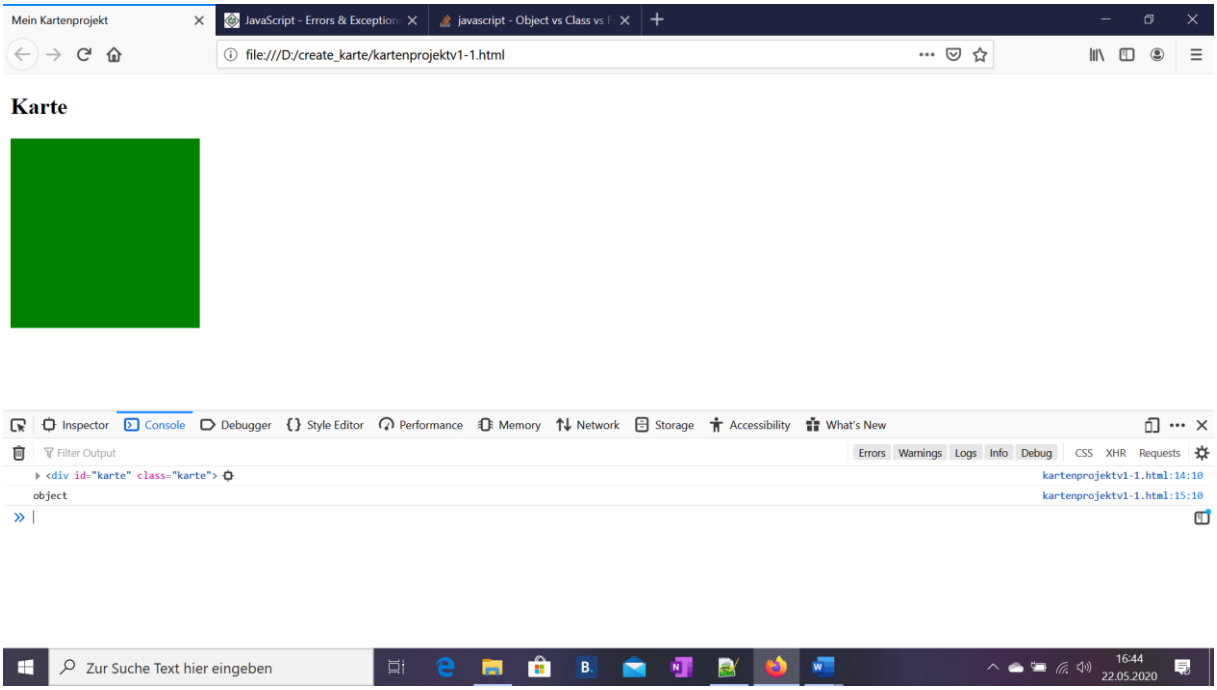
2. **Was muss alles geändert werden, wenn man als Id im div-Tag das englische Wort für karte verwendet, „map“ ?**

Die Karte sieht sehr langweilig aus. Es könnte aber durchaus ein Ausschnitt aus einem Stück Wald sein.

Aufgabe

3. **Wie könnte man einen Ausschnitt aus einem See oder Ozean darstellen?**

Durch Rechtsklick auf das grüne Rechteck -> Element untersuchen, wird ein mächtiges Entwicklungsinstrument im unteren Bereich, oder rechts im Fenster geöffnet.



Zunächst interessiert uns nur, was unter Console oder Konsole angezeigt wird.
Der Befehl console.log() schreibt den Inhalt der Klammern auf die Webseite. In unserem Beispiel wird einmal die Instanz des ganzen div Objekts in der Variablen a ausgegeben und dann der Typ (typeof) der Instanz in der Variable.
Man kann also sehr einfach kontrollieren, ob die Variablen auch den Inhalt haben, den man vermutet.

Aufgabe

4. **Was verbirgt sich hinter a.id? Welchen Typ hat a.id?** Übrigens: Ein String ist eine Zeichenkette.

Aufgabe

5. **Wir fügen unterhalb der var a = Anweisung ein: var b=1; Welchen Typ hat b?**

Aufgabe

6. **Wie nennt man den Listentyp der durch die Zuweisung var liste = [1,2,5]; eingeführt wird?** Dazu einfach „console.log(liste);“ zwischen die script-Tags einfügen.

Jetzt wollen wir aber eine richtige Karte einfügen. Bei unseren bisherigen Beispielen konnten wir Instanzen durch Tags erzeugen. Bei einer Webkarte ist das Objekt nicht im Browser verfügbar. Wir müssen eine Bibliothek einfügen, die das Objekt, Methoden und Eigenschaften zur Verfügung stellt.
Dieses Kartenobjekt wird u.a. von openlayers im Internet in verschiedenen Versionen (v6.x.y mittlerweile) zur Verfügung gestellt unter der Adresse:
https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/build/ol.js

Um diese javascript Programmbibliothek einzufügen, stecken wir das Ganze in script-Tags zwischen die Head-Tags der Seite.
Die Karte soll in einem bestimmten wiedererkennbaren Stil geladen werden, dieser Stil wird ebenso zwischen die Head-Tags geschrieben.

Aufgabe

7. **Erstelle mit „Speichern unter“ aus der Datei kartenprojektv1-1.html eine Datei kartenprojekt_v1-2.html und stelle sicher, dass der Head-Tag Abschnitt so aussieht:**

```
<head>
     <meta charset="utf-8"/>
    <title>Mein Kartenprojekt</title>
     <script
src="https://cdn.rawgit.com/openlayers/openlayers.github.io/ma
ster/en/v5.3.0/build/ol.js"></script>
     <link rel="stylesheet"
href="https://cdn.rawgit.com/openlayers/openlayers.github.io/m
aster/en/v5.3.0/css/ol.css" type="text/css">
  </head>
```

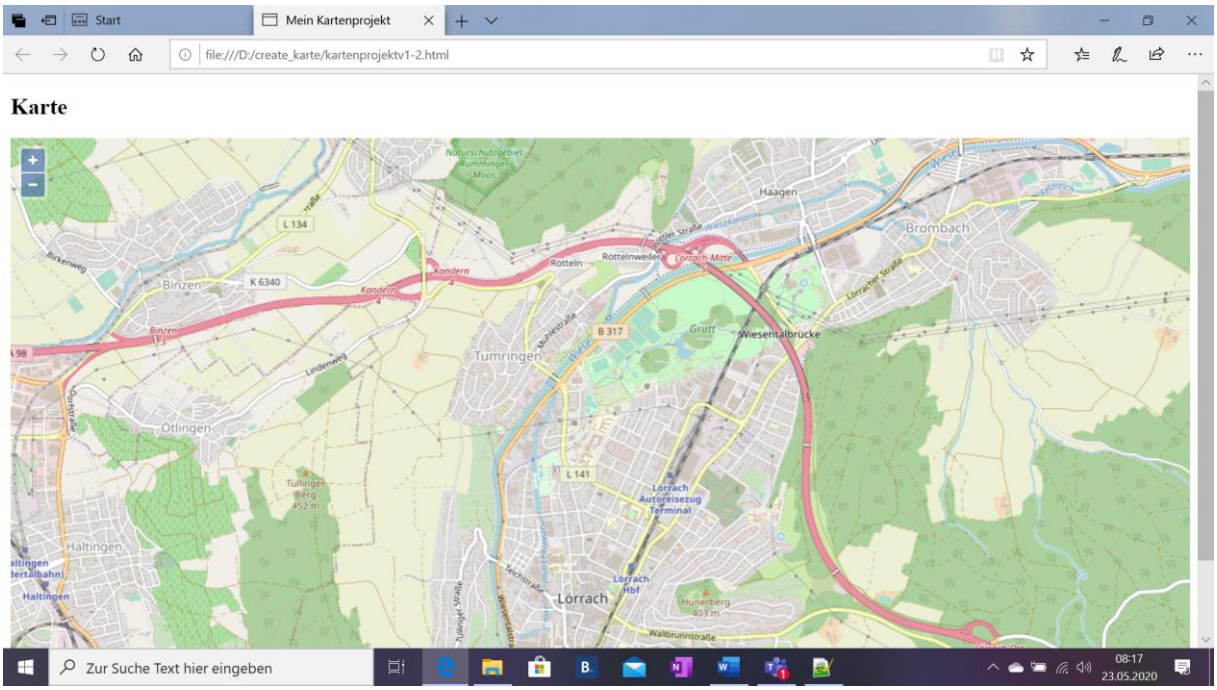
Ersetze anstelle des <script>-Abschnitts im <body>Abschnitt:

```
<script>
var karte = new ol.Map({
        target: 'karte',
        layers: [
          new ol.layer.Tile({
            source: new ol.source.OSM()
          }),
        ],

        view: new ol.View({
          center: ol.proj.fromLonLat([7.662277, 47.622213]),
          zoom: 14
        })
      });
</script>
```

Die gesamte Datei kartenprojekt_v1-2.html sieht jetzt so aus:

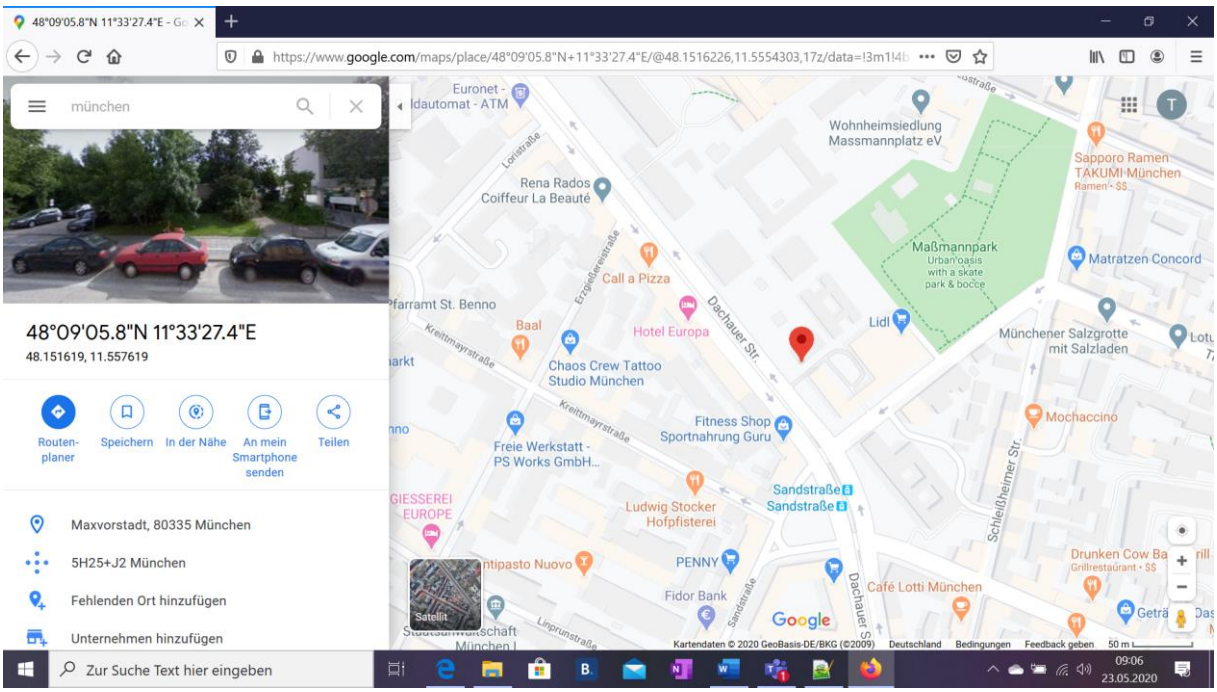
```
<!DOCTYPE html>
<html lang="de">
  <head>
     <meta charset="utf-8"/>
    <title>Mein Kartenprojekt</title>
     <script
src="https://cdn.rawgit.com/openlayers/openlayers.github.io/ma
ster/en/v5.3.0/build/ol.js"></script>
     <link rel="stylesheet"
href="https://cdn.rawgit.com/openlayers/openlayers.github.io/m
aster/en/v5.3.0/css/ol.css" type="text/css">
  </head>
  <body>
    <h2>Karte</h2>
    <div id="karte" class="karte">
</div>
     <script>
     var karte = new ol.Map({
        target: 'karte',
        layers: [
          new ol.layer.Tile({
            source: new ol.source.OSM()
          }),
        ],

        view: new ol.View({
          center: ol.proj.fromLonLat([7.662277, 47.622213]),
          zoom: 14
        })
      });

     </script>
  </body>
</html>
```

Wird die Seite im Browser geladen, sollte sie so aussehen:



Aufgabe

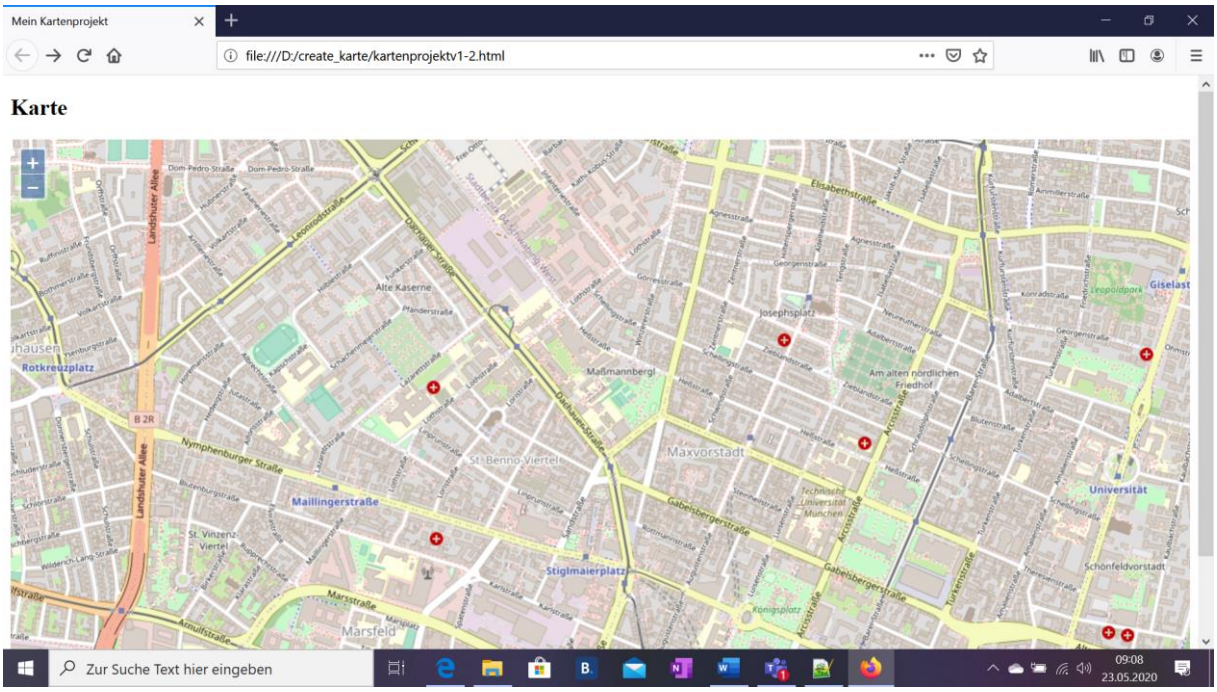
8. **Ändere den Parameter zoom im Abschnitt view im Bereich 4-20. Was ändert sich? Gib Geokoordinaten deines Wohnorts bei center im view an.** Wenn du dich mit Koordinatenreferenzsystemen auskennst, kannst du die EPSG 3857 Koordinaten auch direkt eingeben: center: [Längengrad, Breitengrad] (Deutschland liegt hier ca. center: [1200000, 6500000]), ansonsten einfach in die eckigen Klammern Längengrad, Breitengrad eingeben. Die Daten bekommt man im Internet, z.B. bei google-maps (Rechtsklick auf „Was ist hier“). Leider in der umgekehrten Reihenfolge.



Der Eintrag von z. B.:

```
center: ol.proj.fromLonLat([ 11.557619, 48.151619]),
```

Sollte dann nach Neuladen etwa diese Karte liefern:



Zusammenhang zu Objektorientierter Programmierung:

Um eine Instanz eines neuen HTML-Elementen zu erzeugen, genügt es, Start- und End-Tags ins Dokument zu schreiben. Bei Objekten, die in javascript-Bibliotheken wie der openlayers-Bibliothek definiert werden, muss man

1. den Code zur Konstruktion des Objekts bekannt machen. Wir haben das im head des HTML-Dokuments mithilfe von script-Tags erledigt. Meistens muss man auch noch besondere Stile zugänglich machen.
2. mit Anweisungen wie „var karte = new …“ Instanzen der Objekte erzeugen. Hier ist das Schlüsselwort „new“ wichtig. Bei openlayers muss die Instanz eines Map Objekts die target-Eigenschaft, die Instanz mindestens eines Layer-Objekts und die Instanz eines View Objekts enthalten.

Bei der Instanzerzeugung der Karte werden Eigenschaften und Instanzen neuer Objekte mit übertragen. Im Prinzip erfolgt eine Aufzählung:

```
new ol.Map ({target:“karte“,
```

target: Eigenschaft, die den Ort an dem die Karte dargestellt werden soll, das Ziel „target“, über die Id eines HTML-div-Elements festlegt.

```
new ol.Map ({target:“karte“, layers: [
```

layers: Die eckige Klammer zeigt an, dass eine Aufzählung folgt, es können hier auch drei oder vier Layer (schematische Darstellungen der Geografie, besondere geografische Elemente wie Straßen oder auch Geländeaufnahmen) angegeben werden.

```
new ol.Map ({target:“karte“,
layers: [ new ol.layer.Tile({
```

hier wird die Instanz eines Layers erzeugt, die Zuweisung zu einer Variable ist bereits bei der Karte selbst erfolgt. Man kann allerdings auch einen Layer oder View durch Zuweisung zu einer Variablen erzeugen.

```
new ol.Map ({target:“karte“,
layers: [ new ol.layer.Tile({source: new ol.source.OSM()
```

hier wird dem layer eine Instanz einer Quelle zugewiesen. OSM sind die Kartenobjekte von OpenStreetMap

```
new ol.Map ({target:“karte“,
layers: [ new ol.layer.Tile({source: new ol.source.OSM()})],
```

die geöffneten runde, geschweifte und die eckige Klammer der Layerliste wird geschlossen.
Am Schluss noch das erzeugen der view Instanz:

```
new ol.Map ({target:“karte“,
layers: [ new ol.layer.Tile({source: new ol.source.OSM()})],
view: new ol.View({
```

Hier müssen zwei Eigenschaften (center und zoom) festgelegt werden:

```
center: ol.proj.fromLonLat([ 8.2, 48.151619]),zoom: 8
```

Abschließend müssen alle Klammern geschlossen werden.
}) Klammern des view, }) Klammern des Map Objekts ; Abschluss der Anweisung.

Wem das lieber ist, definiert die layer und view Objektinstanzen **ober**halb der Map-Objektinstanz:

```
var osmlayer = new ol.layer.Tile({…});
var karteView = new.ol.View({…});
```

Die Instanzerstellung der Karte vereinfacht sich dann zu:

```
new ol.Map ({target:“karte“,
layers: [ osmlayer],
view:karteView });
```

Aufgabe

9. **Wie können zwei Karten auf einer Seite dargestellt werden?** Damit die 2. Karte sichtbar wird, sollten die Karten verkleinert dargestellt werden. Wer sich mit css auskennt, kann das mit css lösen, oder wie in am 1. Tag beschrieben, etwas umständlich mit javascript.

Aufgabe

10. **Welche Dateiendungen haben javascript-Dateien, welche style-Dateien? Was ist bei der Einbindung der javascript- und css-Dateien gemeinsam, was unterschiedlich?**

Aufgabe

11. **Wie könnte eine HTML-Seite aufgebaut sein, bei der eine Karte durch Tags eingefügt werden könnte?** Füge dazu ein <geomap> Start-Tag anstelle des Skripts ein. Das ist nur hypothetisch, also beim Speichern aufpassen. Dateiname z. B. html6.html.

Aufgabe:

12. **Beschreibe Gemeinsamkeiten und Unterschiede, wenn man wie unten beschrieben anstelle von openlayers die Bibliothek Chart.js auf der Seite einfügen würde? Chart.js erstellt Diagramme.**

```
<head>
<script src="path/to/chartjs/dist/Chart.js"></script>
</head>
<body>
<canvas id="myChart" width="400" height="400"></canvas>

<script>
var ctx = document.getElementById('myChart');
```

```
var myChart = new Chart(ctx, {
```