“Mont” - an Open Data Mountain Peak Recognition Web App in Vanilla Javascript

Thomas Weibel[1]

[1] *University of Applied Sciences of the Grisons, 7000, Chur, Switzerland*

Abstract
"Mont" is an Augmented Reality web app for iOS/iPadOS devices that displays the names of all the hilltops and mountain peaks in Switzerland, including altitude and distance embedded in the camera stream of a mobile device. The project is based on open data and is written in Vanilla JS, without any third-party libraries or frameworks.

Keywords
HTML5, javascript, vanillaJS, webapp, mobile, open source, open data, open government data, geodata, geoinformatics, topography, Switzerland

1. Introduction

The history of the web, originally proposed in 1989 by Tim Berners-Lee [1] as a concept for a central data repository at "Cern" in Geneva, can be divided into the following development stages:

- Structured documents (HTML, 1990);
- Interactivity, Web 2.0 (server-side programming, Perl/PHP, 1994/1995);
- Graphic design (CSS, 1996);
- Online services and applications (client-side programming, JS, 1996);
- Mobile optimisation (CSS Media Queries, 2000);
- Social media (2003);
- Extended Augmented Reality (VR, AR, 2015).

The collective term "Extended Reality" (XR) stands for "Virtual Reality" (VR), "Augmented Reality" (AR) and "Mixed Reality" (MR) and covers a range of approaches that aim to superimpose data on the reality experienced on suitable devices and thus offer added informational value. Today, it is possible to access the camera and motion sensors of the end device using JavaScript (JS) and thus - at least in theory – create hand-programmed AR web apps. However, most online projects in augmented reality (VR, AR, MR) have so far required 3D software (closed source), graphics engines or the use of relevant JS frameworks.

The question that led to my "Mont" project was therefore: is it possible to develop an Augmented Reality web app for smartphones

- as a hand-written web page (HTML, Javascript),
- without the use of external libraries or frameworks,
- without the use of closed source software,
- on the basis of open data (Open Government Data, OGD)?

thomas.weibel@bluewin.ch (T. Weibel)
0000-0001-9486-7884 (T. Weibel)

5th International Conference on Creative\Media/Technologies (IConCMT), November 28-30, 2023, St. Pölten University of Applied Sciences, Austria

My attempt was to program an AR web app in pure (so-called "vanilla") Javascript. In the technical sense, "Mont" is a website designed for smartphones (mobile only) with only 11 kb of pure program code, which offers a camera view of the surrounding landscape, calculates the local topography and displays the names of all visible hilltops and mountain peaks (including their summit height and distance) within a given range. In addition, a compass indicates the current viewing direction and an altimeter the sea level of the current location.

"Mont" is accessible via a URL [2] and is based on the collection of geographical names from the Swiss National Survey and the digital elevation model of Switzerland, which were published as OGD [3] by the Federal Office of Topography on the opendata.swiss platform. All the necessary data (spatial information from the digital elevation model of Switzerland and all the hilltops and mountain peaks in Switzerland recorded by Swisstopo) are stored in the form of plain text files. As a purely client-side application, "Mont" does not store any user data.

2. The WebApp “Mont”

Augmented Reality essentially requires the following:

- Camera view of the user environment;
- Geolocalisation;
- Access to the gyroscope and acceleration sensors of the end device;
- Access to the compass/magnetic sensor of the end device;
- Data and content to enhance the user experience.

Augmented reality applications generally require specific hardware - dedicated VR or AR glasses equipped with cameras and sensors, smartphone extensions such as the Google Cardboard [4] equipped with optical lenses or even large-scale visualisation environments consisting of complex audio and projector systems. This also results in their limitations: Because these devices are not available in everyday user life, the only potentially AR-capable device that people carry with them at all times is their smartphone. The "Mont" web app is therefore consistently designed for smartphones, should be as easy to use as possible (when walking or hiking, for example) and should not require any additional hardware.

For reasons that will be explained in more detail in the course of this paper, all OS and device-specific information refers to Apple-branded mobile devices and the iOS or iPadOS operating system.

3. Data

On November 30, 2018, the Federal Council of Switzerland adopted the "Open Government Data" strategy for the years 2019-2023, which stipulates that the federal administration will publish its data for free use, with the exception of data where a "legitimate interest in protection" [5] prevails, for example in the areas of public security or data protection.

Since the Federal Council's strategy was published on November 30, 2018, large datasets have been published in several stages - since March 1, 2021, for example, all geodata from the Federal Office of Topography has been freely available. This data is available in a well-documented form on the opendata.swiss platform and can be freely downloaded and used.

3.1. Digital height model of Switzerland

The basic prerequisite for an application in the field of topography is the availability of spatial data that depicts the terrain structure of Switzerland. The digital elevation model DHM 25/200 with a mesh size of 200 meters [6] describes the terrain surface of Switzerland in three dimensions and provides an abstracted terrain model in the form of around 1.5 million data points with their

coordinates in all three spatial axes. The x- and y-axes correspond to the longitude (eastern longitude) and latitude (northern latitude) in meters of the Swiss National Survey of 1903 (LV03) or of 1995 (LV95). The z-axis (spatial axis) is given in meters above sea level. The detailed elevation model [7] is available in various table formats as well as a complete 3D model of Switzerland in Drawing Interchange Format (DXF), which can be used for import into most 3D programs.

3.2. Collection of Swiss geographical names

The names of all terrain elevations in Switzerland originate from the collection of geographical names of the National Survey swissNAMES3D [8]. This dataset represents the most complete toponymic collection in Switzerland (and the Principality of Liechtenstein) and forms the basis for the labelling of all Swiss national maps. For this "Mont" application, all entries in the categories "Alpine summits", "Main summits", "Peaks", "Main hills" and "Hills" were taken from the database. This means that the "Mont" web app is able to recognise and display over 9000 elevations in Switzerland.

3.3. Storage and retrieval

"Mont" does not store the elevation data of the terrain model and the terrain elevations in a database, but in file form as JS arrays. These arrays with the variables `a, b` and `c` (longitude, latitude and sea level) for the elevation model and the variables `nm, lo, la` and `he` (summit names, longitude, latitude and summit height) are integrated at the beginning of the HTML document as external scripts - the geographical longitude of all terrain points of the elevation model, for example, as `<script src="data/a.js"></script>` - and declared as variables in the actual `script.js` program script using `var`. Using the "hoisting" method [9], with which JavaScript places declarations of functions, variables or classes at the head of the program code to be executed by default, all spatial information can be queried and processed in the further course of the script in the syntax variable `name[index number]`. Terrain points of the elevation model are thus available as $a[n_1]$, $b[n_1]$ and $c[n_1]$ according to the spatial axes; summit names as $nm[n_2]$, $lo[n_2]$, $la[n_2]$ and $he[n_2]$.

4. Programming

Because "Mont" is a website in the technical sense, no app installation is required to use it. After access, the necessary metadata, the application data, the styling (CSS), the markup (HTML) and finally the program (JS) are loaded and executed (in this order). Depending on the configuration made, the smartphone requests consent for

- Geolocalisation,
- access to the camera,
- and access to the gyroscope sensors.

To ensure functionality, consent is required for all three points; in the event of a refusal, "Mont" issues a corresponding error message. With the exception of the position sensors or the gyroscope and acceleration sensor, consent can be saved in the browser settings so that the corresponding queries are not made the next time the web app is used.

As soon as "Mont" is displayed on the screen, the web app is ready for use. Because all data is cached in the browser cache, the application remains functional even offline, for example in rough terrain without signal, and does not need to be reloaded to be used again.

Once all the necessary consents have been obtained, the app locates the device and then converts the coordinates received from the GPS sensor into Swiss national coordinates. The digital elevation model is then used to determine the sea level of the four nearest spatial points, and the approximate sea level of the current user location is calculated using bilinear interpolation. The app then searches the coordinates and sea levels of all mountain peaks and hilltops in Switzerland, filters out elevations

within a range of 20 km and uses horizontal and vertical triangulation to calculate all those that are actually visible from the current location. The name, altitude and distance are added above each peak visible on the smartphone screen. The font size of the summit names displayed depends on the respective airline distance; all labels follow the respective summits across the screen according to the movements of the smartphone.

A compass at the bottom left shows the orientation of the smartphone on an analogue compass rose and in compass degrees; an altimeter at the bottom right shows the interpolated sea level numerically and analogously on a 1000 m scale. Finally, the refresh button allows the topography to be recalculated after a change of location.

4.1. Mode of operation

In technical terms, "Mont" is a web page in HTML 5 with an external stylesheet in CSS 3, external data arrays and the external program code in Javascript ES 6. To use the web app, it is sufficient to load the page completely.

4.1.1. Camera stream

While conventional websites display text and media content on a neutral background, an AR web app requires access to the video stream from the device camera, which can then be provided with the desired information. Because most mobile devices have at least one camera, the World Wide Web Consortium (W3C) first proposed the JS method `getUserMedia` of the `navigator` object [10], the simplest form of which is

```
const stream=await navigator.mediaDevices.getUserMedia({
  video: {
    width: 1280,
    height: 720,
    aspectRatio: 16/9
  }
});
```

and which can already programmatically access the video stream from the main camera and embed it in a website. Text and media content can then be displayed in the form of overlays over this dynamic view of the real user environment.

4.1.2. Geolocation

A prerequisite for applications that display location-dependent data is the most precise possible localisation of the current user location. The geolocation API has been part of the HTML5 standard since 2008 [11]. In addition to the position, the accuracy of the coordinates, the speed of movement and, at least theoretically, the direction of movement can also be queried. As determining the location is a major invasion of the user's privacy [12], queries are only possible in secure contexts, i.e. only via the HTTPS protocol, and each location query requires the express consent of the user (at least for the first time).

Geolocation determines the current latitude and longitude using the GPS sensor (if available) or the mobile phone signal, accessible WLAN networks and the IP address of the device (if GPS is not available) [12]. If the IP address is used, the data of the provider exchange node is used to determine the position.

The `longitude` and `latitude` properties can be queried using the JavaScript method `getCurrentPosition()` of the `navigator` object. Outdoors and depending on the availability of GPS on the end device or the quality of the mobile phone/WIFI signals received, HTML5 geolocalisation achieves accuracies of 5-20 m. The output is in decimal degrees according to the WGS84 geodetic reference system. For the Chur media center, for example, this information is 46.84793 degrees north latitude and 9.50186 degrees east longitude. However, because the geodata of the Federal Office of Topography is available in the formats of the Swiss national survey of 1903 or 1995, the decimal

degrees according to WGS84 must be converted into meters according to LV03 or LV95. "Mont" uses the national coordinates (longitude as variable e, latitude as variable n) in LV03 format.

```
navigator.geolocation.getCurrentPosition((position)=> {
  let b1=position.coords.latitude*3600;
  let l1=position.coords.longitude*3600;
  let b2=(b1-169028.66)/10000;
  let l2=(l1-26782.5)/10000;
  let n=200147.07+308807.95*b2+3745.25*Math.pow(l2,2)+76.63
*Math.pow(b2,2)+119.79*Math.pow(b2,3)-194.56*Math.pow(l2,2)*b2;
  let e=600072.37+211455.93*l2-10938.51*l2*b2-0.36*l2 *Math.pow(b2,2)-
44.54*Math.pow(l2,3);
}
```

The corresponding coordinates for the above-mentioned Medienhaus Chur site are 757,348 m east longitude and 190,601 m north latitude.

4.1.3. Motion sensors

Position/acceleration sensors and gyroscope enable the spatial orientation of the device to be read out along the three spatial axes `x, y` and `z` using the event listener `deviceorientation` and in the three position angles `alpha, beta` and `gamma` in accordance with the W3C specification [11].

The query

```
if (typeof DeviceMotionEvent.requestPermission==="function") {
  DeviceMotionEvent.requestPermission()
    .then((state)=> {
      if (state==="granted") {
        window.addEventListener("deviceorientation", direction);
      }
      else {
        fail();
      }
    })
  .catch(console.error);
}
else {
  window.addEventListener("deviceorientation", direction);
}
```

can then be used to determine the position angle using the JS properties `event.alpha, event.beta` and `event.gamma,` regardless of the iOS or iPadOS version used, and to trigger the `direction` function, the procedure for which is explained below.

4.1.4. Magnet sensor

For horizontal triangulation, the (experimental) property `event.webkitCompassHeading` is used in the `direction` function. In contrast to the relative position angle of the vertical axis alpha, which calibrates the zero point ("north") to the orientation of the smartphone when the app is started (so-called "game-based calibration"), the `webkitCompassHeading` property accesses the magnetic compass module installed in the smartphone or tablet and provides absolute compass data (so-called "world-based calibration") [13], which is necessary for correct horizontal triangulation. In addition, the position angle of the longitudinal axis, `event.beta,` is also determined for the purpose of vertical triangulation. Both position angles are stored in the variables `heading` and `tilt.`

```
const direction=()=> {
  heading=event.webkitCompassHeading;
  tilt=event.beta;
  document.querySelector("#needle1").style.transform="rotate("+heading+"deg)";
  document.querySelector("#degrees").innerText=Math.round(heading)+"°";
};
```

Every time the smartphone sensor detects a change in position, `webkitCompassHeading` accesses the magnetic sensor and returns the compass angle. The compass needle `needle1` is then rotated by the new value and the rounded number of degrees is output.

Although the `webkitCompassHeading` property provides absolute alignment data, the values output are generally not very precise. Technical comparative studies show that the deviations between the compass display of a smartphone or tablet on the one hand and a magnetic compass commonly used in geodesy and cartography on the other can be considerable [14]. Particularly in the case of large distances, however, a correct fit between the summit view and the data display would require compass accuracies that only scientific bussoles are capable of delivering. In order to avoid overly disturbing false indications, the detection radius (as a global variable `reach`) was therefore limited to 20 km in the "Mont" web app.

4.1.5. Height interpolation

Correct vertical triangulation requires the most precise possible indication of the current altitude above sea level. To calculate this, the `altimeter` function is used with a three-step bilinear height interpolation.

The variables `h1, h2, h3` and `h4` mark the quadrant in which the user is currently located. Arrays `a` and `b` contain the geographical longitudes and latitudes of all terrain points of the digital elevation model, array `c` contains the respective sea levels. The variables `fx` and `fy` denote the relative distance to the next lower measuring point on the x-axis (longitude) and y-axis (latitude) respectively.

Linear interpolation is applied a total of three times, first twice for the x-axis (the respective roundings using `Math.floor` refer to the DHM grid width of 200 m) and then once for the y-axis.

```
const altimeter=()=> {
  let h1, h2, h3, h4;
  for (i=0; i<a.length; i++) {
    if (a[i]==Math.floor(locX/200)*200 && b[i]==Math.floor(locY/200)*200)
h1=c[i];
    if (a[i]==Math.ceil(locX/200)*200 && b[i]==Math.floor(locY/200)*200) h2=c[i];
    if (a[i]==Math.floor(locX/200)*200 && b[i]==Math.ceil(locY/200)*200) h3=c[i];
    if (a[i]==Math.ceil(locX/200)*200 && b[i]==Math.ceil(locY/200)*200) h4=c[i];
  }
  let fx=(locX-Math.floor(locX/200)*200)/200;
  let fy=(locY-Math.floor(locY/200)*200)/200;
  let ht1=fx*(h2-h1)+h1;
  let ht2=fx*(h4-h3)+h3;
  let ht=fy*(ht2-ht1)+ht1;
};
```

`ht1` is determined from `h1` and `h2` using linear 1D interpolation. Subsequently, `ht2` is determined in the same way from `h3` and `h4`. Finally, the bilinear height `ht` determined in this way results from a linear 1D interpolation between `ht1` and `ht2`.

4.1.6. Peak triangulation

After locating the device, determining its horizontal and vertical position angles and the interpolated sea level, all the information required for successful triangulation of all visible mountain peaks and hilltops is now available - together with the height information from the digital elevation model and the collection of geographical names from the national survey.

`Math.sqrt(Math.pow(nodeX,2)+Math.pow(nodeY,2))` is used to determine the straight-line distances between the user position and all DHM grid points (whereby the variables `nodeX` and `nodeY` represent the difference in geographical longitude and latitude and therefore the length of the two sides of the corresponding right-angled triangle) and stored temporarily in the auxiliary array `nodeDists`. The arcsine from the ratio of the latitude difference `nodeY` and the grid point distance `nodeDist`, multiplied by 180/π, results in the horizontal position angle `nodeHAng` East as an absolute sum of 90 degrees for positive latitude differences and the horizontal position angle `nodeHAng` West as an absolute difference of 270 degrees for negative latitude differences.

```
if (nodeX>0) nodeDistAng=90+Math.asin(nodeY/nodeDist)*180/Math.PI;
else nodeDistAng=270-Math.asin(nodeY/nodeDist)*180/Math.PI;
```

```
let nodeHAng=Math.abs(nodeDistAng);
```

The absolute vertical position angle can be determined analogously as an arctangent from the ratio of height difference `c[n]-ht` and grid point distance `nodeDist`, multiplied by 180/π:

```
Math.abs(Math.atan((c[i]-ht)/nodeDist)*180/Math.PI);
```

The horizontal and vertical position angles of all topographical elevations are then determined in the same way. Finally, the index numbers of all grid points of the elevation model as well as all hilltops and mountain peaks that lie within the recognition distance of 20 km are temporarily stored in the auxiliary arrays `nodes` and `peaks` respectively. The exact compass direction and inclination relative to the user's position is now available for each grid point of the digital elevation model and for each peak within the recognition radius.

4.1.7. Peak visibility

It would not make sense to output all the summit information available within a visual range of 20 km because only a few of them are actually visible from the user's point of view. For this reason, the next step is to evaluate all available spatial information from the DHM and the collection of geographical names.

The mathematical assessment of the visibility of a peak lying within the detection distance can be achieved with a nested loop, which compares all peak heights of the indices from `peaks` with the sea level heights of all DHM grid points of the indices from `nodes` lying between user and mountain position (within a horizontal and vertical tolerance). To take account of the horizontal extent of hills and mountains, the horizontal tolerance (as global variable `toleranceX`) is 20 degrees and the vertical tolerance (as global variable `toleranceY`) is 1 degree.

```
for (i=0; i<peaks.length; i++) {
      for (j=0; j<nodes.length; j++) {
    if (
      Math.abs(nodeAngX[nodes[j]]-peakAngX[peaks[i]])<toleranceX &&
      nodeDists[nodes[j]]<peakDists[peaks[i]] &&
      nodeAngY[nodes[j]]>peakAngY[peaks[i]]+toleranceY
    ) peaks[i]=null;
  }
}
```

If terrain points located within the horizontal 20-degree sector in front of a mountain peak have higher vertical elevation angles, they obscure the view and the peak in question is marked as a `null` object and thus filtered out of the display. In order to use all available spatial information (and thus to take into account not only the sea level heights of all DEM grid points, but also the summit heights of all topographic elevations), the nested loop is repeated analogously with the indices of `peaks` lying within the detection radius. Apart from algorithm-related imponderables (compass deviations, inaccuracy of bilinear height interpolation, etc.), this method reliably filters out non-visible peaks.

4.1.8. Display

The result is displayed on the smartphone screen using a loop that adds all visible mountain peaks indexed in the `peaks` array from the peak array `nm` to the DOM (or the parent element `canvas`) as a `<p>` tag of the CSS class `output` and provides additional information (aerial line distance from `peakDists` and peak height from `he`). The font size is inversely proportional to the distance, except for a minimum size of 2em; a `z-index` overlays the labels of closer peaks on more distant ones. The styling of the labels in turn depends on the determined font size.

```
for (i=0; i<peaks.length; i++) {
  const para=document.createElement("p");
  canvas.appendChild(para);
  para.id="o"+i;
  para.className="output";
  para.innerHTML=nm[peaks[i]]+" <span class=\"small\"> "+he[peaks[i]]+"m
"+Math.round(peakDists[peaks[i]]/100)/10+"km<\/span>";
  para.style.fontSize=5-3*peakDists[peaks[i]]/reach+"em";
  para.style.zIndex=Math.round(reach-peakDists[peaks[i]])+2;
```

```
    para.style.borderRadius=para.offsetHeight/2+"px";
  }
```

4.1.9. Animation

Finally, the animation needs to be displayed so that all labels move across the screen in line with the horizon when the end device moves horizontally or vertically. Positioning and triangulation are carried out in three dimensions throughout, while the screen animation of the summit information is carried out in two dimensions and using linear CSS transitions.

In order to register the individual summit names on the horizon, a loop is added to the aforementioned compass function `direction`, which is called each time the horizontal position changes, which runs through all the summits within recognition range and repositions them on the screen. The factor `fov` determines the ratio of the (horizontal and vertical) animation to any panning movements by the user. This factor is determined by the focal length of the rear camera, which is 26 mm on Apple smartphones and tablets [15], which corresponds to a field of view (FOV) of 69.39 degrees in landscape mode [16]. However, since the "Mont" web app is used in the vertical position typical for smartphones (portrait mode) and the camera sensor has an aspect ratio of 16:9, this results in a proportional vertical FOV of 39.09 degrees, which in turn corresponds to the full screen width of `100vw` and thus results in an animation factor of 2.562 `vw`/degree. Multiplying the compass orientation `heading` by the factor `fov` plus 50 therefore places the labels horizontally in the center of the screen according to their direction; multiplying the `tilt` angle tilt by `fov` plus 89 (as a proportional result of half the screen width 50/9*16) centers the labels vertically.

```
  let fov=2.562; // 26mm focal length > 69.39° fov horizontal, 16:9 ratio > 39.03
fov vertical = 100vw
  for (i=0; i<peaks.length; i++) {
    let label=document.querySelector("#o"+i);
    label.style.left=((peakAngX[peaks[i]]-heading)*fov+50)+"vw";
    label.style.bottom=((peakAngY[peaks[i]]+90-tilt)*fov+89)+"vw";
  }
```

Other smaller routines fix inconsistencies in the display such as the dead angle (resulting from the periodisation of the display) when the compass passes through zero, incorrect display of the labels when the compass passes through 270, 0 and 90 degrees due to the linear CSS transition or a numerical compass display of 360 instead of 0 degrees.

4.2. Error handling

Because errors can occur when using "Mont" (missing authorisations of the web browser, use outside the data area, etc.), app-wide error management with the output of relevant error messages is necessary. A distinction is made between two types of error: incorrect orientation of the smartphone and errors that require the web app to be reloaded.

4.2.1. Smartphone orientation

In order to keep the use of "Mont" as intuitive as possible, it is intended to be used in the portrait mode typical of smartphones. Because the landscape mode affects the calibration of the magnetic compass module - turning the iPhone leads to a deviation of 90 or -90 degrees - a simple CSS media query is used to display a graphical error message `<mark>` that hides the user interface `<main>`.

```
  @media (orientation:landscape) {
    main {
      visibility: hidden;
    }
    mark {
          visibility: visible;
    }
  }
  @media (orientation:portrait) {
    main {
```

```
        visibility: visible;
    }
    mark {
        visibility: hidden;
    }
}
```

The graphic shown in `<mark>` prompts the user to hold the smartphone in portrait format. After turning it to the correct position, `<mark>` disappears and the app is displayed.

4.2.2. Other

The following errors prevent the correct functioning of "Mont" and may require a change to the device settings. Errors of this type are added to the `msgs` message array, which is checked by the `fail` function when the app is started. If there are messages (or if the length of `msgs` is >0), the messages are listed, which prevents the UI from being displayed.

```
const fail=()=> {
  if (msgs.length>0) {
    let msg=msgs.join("\n");
        document.querySelector("#errorType").innerText=msg;
    document.querySelector("#error").style.visibility="visible";
  }
};
```

If the error can be rectified (e.g. if the user has not given their consent), "Mont" must then be reloaded.

4.2.3. Operating system

Because the `webkitcompassheading` property used to determine the horizontal alignment is currently only supported by Apple mobile devices [17],"Mont" checks the operating system at startup and displays the corresponding message on devices with Android or Windows Phone. The topographical calculation is only started on Apple devices (iPhone, iPad, iPod) using the `query` function.

```
const getMobileOS=()=> {
  let userAgent=navigator.userAgent || navigator.vendor || window.opera;
  if (/windows phone/i.test(userAgent)) {
    msgs.push("Kein iOS-Gerät");
        console.error("Kein iOS-Gerät");
        fail();
  }
  if (/android/i.test(userAgent)) {
        msgs.push("Kein iOS-Gerät");
        console.error("Kein iOS-Gerät");
        fail();
  }
  if (/iPad|iPhone|iPod/.test(userAgent) && !window.MSStream) {
        query();
  }
};
```

4.2.4. Geolocation

In order to prevent possible tracking by websites, many users do not grant the web browser permission for localisation. Either access to localisation data via GPS (or accessible mobile phone signals, WLAN networks and the IP address of the device) is switched off device-wide, or the request for the corresponding authorisation is rejected when the app is started. In this case, localisation is impossible and therefore the prerequisite for summit triangulation is not met, which is acknowledged with the corresponding message.

```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition((position)=> {
    // geolocation success
```

```
    }
    altimeter();
    }, ()=> { // geolocation fail
      msgs.push("Ortung nicht möglich");
      fail();
    }, { // geolocation options
      enableHighAccuracy: true,
      timeout: 5000,
      maximumAge: 0
        });
  }
  else {
        msgs.push("Ortung nicht möglich");
    fail();
  }
```

4.2.5. Geography

"Mont" only contains topography data for the Swiss national territory. Although it would be possible to use it abroad by setting a default location (such as the old Bern Observatory, the "zero point" of the Swiss coordinate system) [18], this is not practical for an AR application based on the real user environment. For this reason, the border coordinates are queried to prevent the UI display in the event of access from a location outside Switzerland.

```
  if (e>=480000 && e<=865000 && n>=74000 && n<=302000) {
    locX=e;
    locY=n;
    init();
  }
      else {
    msgs.push("Ausserhalb der Schweiz");
    fail();
    init();
  }
```

If a call is made outside these limit coordinates, a corresponding error message is also issued.

4.2.6. Camera

Like any AR application, "Mont" requires access to the camera stream. If this is prevented by the user's lack of consent, the prerequisite for the web app to function is not met and the UI is not displayed.

```
  if (navigator.mediaDevices.getUserMedia) {
    navigator.mediaDevices.getUserMedia({video:{facingMode: "environment", height:
2560, width: 1440, aspectRatio: 16/9}})
    .then ((stream)=> {
      if (initial) document.querySelector("#camera").srcObject=stream;
      initial=false;
        })
        .catch ((err)=> {
      msgs.push("Kein Zugriff auf Kamera");
      fail();
        });
  }
  else {
    msgs.push("Kein Zugriff auf Kamera");
        fail();
  }
```

If access to the camera is successful, the output on the screen is formatted according to the device specifications in terms of height, width and aspect ratio and the stream is initialised.

4.2.7. Magnet compass, gyroscope

Correct registration of the peak labels on the horizon and animation in line with the device movements requires access to the magnetic compass module built into the device as well as the position and acceleration sensors. In contrast to localisation and camera access, since the release of iOS 13 (September 1, 2020), consent can no longer be granted via the system settings but requires user interaction according to Apple design guidelines [19]. The `activate` start function is therefore part of the `menu` function, which must be activated by the user with a click to close the app intro.

```
const menu=()=> {
  const instr=document.querySelector("#info");
  if (instr.style.visibility=="hidden") instr.style.visibility="visible";
  else {
    instr.style.visibility="hidden";
        activate();
  }
};
```

If the subsequent authorisation request for access to the motion sensors is rejected, correct registration of the labels on the horizon is not possible and the UI is hidden.

4.2.8. Properties

Like all web apps, "Mont" is extremely easy to access: all you need to do to use the application is click on a web link, use a QR code pointing to it or enter the URL. Because the web app consists exclusively of client-side web technologies, the source code of "Mont" is openly accessible; the app is also based on open data, which is also available under an open license [20]. The only condition for the use and publication of the geodata is that the source is cited in accordance with the terms of use of the Federal Office of Topography.

As a pure web project, "Mont" is extremely data-efficient: the total amount of data is only 31 MB; the size of the actual program code is only 11.1 kb. Because no external libraries or frameworks are used, the web app is self-sufficient and therefore easy to port and adapt. Programming is efficient and transparent for developers thanks to inline commenting.

Because "Mont" only uses freely editable plain text files for programming and data and does not have any external dependencies thanks to the absence of libraries and frameworks, the app can be adapted to any task with little effort. The only requirement is the availability of georeferenced data from which the data arrays required for "Mont" can be generated.

4.3. Limitations

The biggest limitation when using "Mont" is the lack of compatibility with Android or other operating systems. Although the web app is based entirely on the open web technologies HTML 5, CSS 3 and JS ES 6, the W3C has been proposing a standard for reading the magnetic compass module installed in smartphones since 2021 [20], the corresponding JS object `Magnetometer` is still not compatible with most web browsers. As a robust JS method for reading the compass orientation, i.e. suitable for all platforms, is still missing and the `webkitcompassheading` property used for "Mont" is also not part of the web standard, "Mont" remains experimental. For this reason, it currently only makes sense to use it in environments in which the hardware environment can be controlled (e.g. through a device handover by companies or institutions).

In addition, the magnetic compass module built into smartphones is not very reliable. Even away from sources of magnetic interference (inside buildings, in Faraday cages such as cars, near overhead power lines, etc.), deviations from a conventional magnetic compass can be detected, which impairs the accuracy of information registration on the horizon. In the course of ongoing hardware optimisations, however, an increase in performance can also be expected in the area of sensor technology (magnetic compass, gyroscope, acceleration sensors) in current smartphone models.

The handling of "Mont" as a website, which requires extensive authorisations (localisation, camera and sensor access), does not meet the expectations of native apps in terms of UX. Even if access to the camera stream and gyroscope data can be regulated system-wide in the settings, the handling of the various rights queries by the user remains cumbersome. This restriction results from the system specifications of the operating system and browser manufacturers and cannot be remedied by the web app. However, as more and more powerful web apps are being launched on the market in the geoinformatics sector, it is to be expected that future OS and browser versions will simplify the relevant rights management.

The use of "Mont" is ultimately limited to the Swiss national borders because only OGD from the Swiss authorities was used. However, it is also possible to adapt the web app for purposes abroad without further ado if georeferenced data in WGS84 format is available.

4.4. Further development

The development of the "Mont" web app revealed a number of insights into the performance of current web technologies that were previously poorly documented. However, due to the limitations mentioned (lack of platform independence, limited UX), the commercial use of "Mont" is not currently viable. One exception could be its use in a closed hardware environment in which the required iOS end devices are handed over to users. Applications in tourism, culture or the public sector that want to provide location-based information (historical information on buildings and places within an urban area; tourist infrastructure or botanical/zoological information in a national park, hiking or skiing area; provider or program information at a large festival or trade fair site, etc.) could be considered here. It would make sense to offer the web app in a kiosk-mode browser application that does not allow closing and navigating to other websites.

Existing solutions implemented as native apps for both platforms in the field of geography (such as the official swisstopo app from the Federal Office of Topography) [22] or topography (such as PeakFinder from PeakFinder GmbH, Zurich) [23] have become well established on the software market. The strength of the "Mont" web app lies in the fact that the type of information conveyed can be adapted to any purpose with little effort: Any type of georeferenced content can be displayed by simply exchanging the corresponding data arrays (and, depending on the content type, by generating the corresponding markup). Any type of information (text, audio, graphics, images and video) can be used as content. In this way, as a prototype application or as an offer in a closed hardware environment, multimedia AR guides can be realised cost-effectively and within a short period of time.

A fundamentally different option is to implement it as a native application for the iOS and Android platforms. Native programming would eliminate the majority of the limitations mentioned but would result in a substantially larger project scope and therefore additional time and financial outlay.

5. Conclusions

The programming of the "Mont" web app can be described as a successful failure: The lack of robust JS methods for determining absolute alignment data and the resulting forced reliance on the experimental `webkitcompassheading` property prevents the creation of geodata and AR web apps optimised for all common platforms. Nevertheless, the present experiment has shown that today's web technologies are far more powerful than websites generally suggest. Augmented reality applications can also be realised without the help of external JS libraries or frameworks.

The W3 consortium recognised the importance of a compass function for the creation of geodata-based applications at an early stage. As early as 2009, W3C authors proposed methods [24] with which - in addition to geolocation - device orientation in absolute compass degrees should also have been possible. Although the current version of the W3C's Geolocation API [25] provides various options for querying the compass orientation, these have not yet been implemented in the case of the `heading` attribute as proposed by the W3C.

Due to the experimental method for determining the compass orientation and the aforementioned UX deficits, the "Mont" web app only has experimental status. However, future developments in the field of web technologies and sensor technology could provide a remedy so that robust, platform-independent programming of extended reality applications in the form of web apps in vanilla JS becomes possible.

References

[1] T. Berners-Lee, Information management: a proposal, 1989. URL: https://www.w3.org/History/1989/proposal.html.

[2] T. Weibel, „Mont“, 2022. URL: https://www.thomasweibel.ch/mont2/.

[3] Swisstopo, Digitale Geodaten von swisstopo sind neu kostenlos und können frei genutzt werden, 2021. Bern. URL: https://www.swisstopo.admin.ch/de/home/meta/medieninformationen.detail.news.html/swisstopo-internet/news2021/news_release/20210301.html.

[4] Google, Google Cardboard, 2014. URL: https://arvr.google.com/cardboard/.

[5] Schweizerische Eidgenossenschaft, Strategie für offene Verwaltungsdaten in der Schweiz 2019–2023 (Open-Government-Data-Strategie, OGD-Strategie), 2018, p. 2. URL: https://www.newsd.admin.ch/newsd/message/attachments/55083.pdf.

[6] Swisstopo, DHM 25/200, das digitale Höhenmodell der Schweiz mit einer Maschenweite von 200 m, 2010. URL: https://opendata.swiss/de/dataset/das-digitale-hohenmodell-der-schweiz-mit-einer-maschenweite-von-200-m.

[7] Swisstopo, DHM 25/200, das digitale Höhenmodell der Schweiz mit einer Maschenweite von 200 m. Weiterführende Informationen, Dokumente, 2023. URL: https://www.swisstopo.admin.ch/de/geodata/height/dhm25.html#dokumente.

[8] Swisstopo, swissNAMES3D, geografische Namen der Landesvermessung, 2014. URL: https://opendata.swiss/de/dataset/swissnames3d-geografische-namen-der-landesvermessung.

[9] World Wide Web Consortium, The Compass API, W3C Editor’s Draft 29 October 2009, 2009. URL: https://dev.w3.org/2009/dap/system-info/compass.html.

[10] World Wide Web Consortium, Media Capture and Streams, W3C Working Draft 28 June 2012, 2012. URL: https://www.w3.org/TR/2012/WD-mediacapture-streams-20120628/.

[11] World Wide Web Consortium, Geolocation API Specification, Geopriv Suggestion Draft, 3 December 2008. URL: https://www.w3.org/2008/geolocation/drafts/API/spec-source-CDT.html.

[12] World Wide Web Consortium, Geolocation API, W3C Recommendation 01 September 2022, 2022. URL: https://www.w3.org/TR/geolocation/#security.

[13] R. Tibbett, Device Orientation «Alpha» Calibration. Implementation Status and Challenges, 2014, p. 3. URL: https://www.w3.org/2008/geolocation/wiki/images/e/e0/Device_Orientation_%27alpha%27_Calibration-_Implementation_Status_and_Challenges.pdf.

[14] L. Novakova, T. L. Pavlis, Assessment of the precision of smartphones and tablets for measurement of planar orientations: A case study, J. Struct. Geol. 97 (2017) 93ff.

[15] Apple, iPhone 14, iPhone 14 Plus, Tech Specs, 2022. URL: https://www.apple.com/iphone-14/specs/.

[16] W. Fulton, Field of View Calculator (FoV) of a Camera and Lens, 2014. URL: https://www.scantips.com/lights/fieldofview.html.

[17] Mozilla Foundation, Resources for Developers, MDN Web Docs Glossary: Definition of Web-related Terms, Hoisting, 2022. URL: https://developer.mozilla.org/en-US/docs/Glossary/Hoisting.

[18] Swisstopo, Schweizer Koordinatensystem, 2023. URL: https://www.swisstopo.admin.ch/de/wissen-fakten/geodaesie-vermessung/koordinaten/schweizer-koordinaten.html.

[19] Apple, Gyroscope and accelerometer. On-device gyroscopes and accelerometers can supply data about a device's movement in the physical world, 2023. URL: https://developer.apple.com/design/human-interface-guidelines/inputs/gyro-and-accelerometer/.
[20] Swisstopo, Nutzungsbedingungen für kostenlose Geodaten und Geodienste (OGD) von swisstopo, 2021. URL: https://www.swisstopo.admin.ch/de/home/meta/konditionen/geodaten/ogd.html.
[21] World Wide Web Consortium, Magnetometer, W3C Working Draft 7 December 2021, 2021. URL: https://www.w3.org/TR/magnetometer/.
[22] Swisstopo, swisstopo-App, 2020. URL: https://www.swisstopo.admin.ch/de/karten-daten-online/karten-geodaten-online/swisstopo-app.html.
[23] PeakFinder, PeakFinder App, 2010. URL: https://www.peakfinder.org/de/mobile/.
[24] World Wide Web Consortium, The Compass API, W3C Editor’s Draft 29 October 2009, 2009. URL: https://dev.w3.org/2009/dap/system-info/compass.html.
[25] World Wide Web Consortium, Geolocation API, W3C Recommendation 01 September 2022, 2022. URL: https://www.w3.org/TR/geolocation/#introduction.