Anmerkungen

buildCalendar ist deutlich zu lang. Da müssen die Alarmglocken angehen, wenn eine Funktion so lang wird.

Nutze die Möglichkeiten, Code optisch hübsch zu halten.

```
    let howManydaysMonth = lastOfMonth.getDate();
    if(firstWeekDayMonth == 0){ // 0 is Sunday
        daysToDrawBefore = 6; // for the Sunday 6 days to draw
    } else {
        daysToDrawBefore = firstWeekDayMonth -1; // else days to draw before
Monday
    }
    if (lastWeekdayMonth == 0){ //0 is Sunday
        dayToDrawAfter = 0;  // for Sunday no Days to draw
    } else {
        dayToDrawAfter = 7- lastWeekdayMonth // else days to draw to Sunday
    }
    let daysToDraw = daysToDrawBefore + howManydaysMonth + dayToDrawAfter;
```

Hier würde eine Leerzeile zwischen den ifs helfen. Zugegeben, in Dart würde das nicht gehen, außer man bescheißt mit Kommentaren. 😀

Einheitlicher Aufbau hilft beim Lesen .. Vergleiche die folgenden Zeilen:

```
    if(firstWeekDayMonth == 0){ // 0 is Sunday
    } else {
    if (lastWeekdayMonth == 0){ //0 is Sunday
```

und vergleiche

```
    if (firstWeekDayMonth == 0) {   // 0: Sunday
    } else {
    if (lastWeekdayMonth == 0) {    // 0: Sunday
```

Das ist für mich deutlich unaufgeregter im Layout und damit hilft es beim Lesen.

Du kannst mehrere Klassen vergeben.

```
    tablehtml += '<th class="headTable">Fre</th>';
    tablehtml += '<th class="Saturday">Sam</th>';
```

würde bei mir entweder zu

```
    tablehtml += '<th class="headTable Friday">Fre</th>';
    tablehtml += '<th class="headTable Saturday">Sam</th>';
```

oder, weil überall "headTable" (was eh kein guter Name ist) nicht nützlich ist, denn ich kann die Elemente ja eh als `<th>` ansprechen, gar zu

```
    tablehtml += '<th class="Friday">Fre</th>';
    tablehtml += '<th class="Saturday">Sam</th>';
```

```
    /*'Kalenderwoche ausrechnen'um die Kalenderwoche zuermitterln müssen wir den
1.1.xxxx definieren und mit einer schleife abfragen bis wir den ersten Montag im
Jahre gefunden haben.das entspricht dann KW1.
    In einer weiteren berechnung ermitterln wir den wert von Kw1 bis heute und
rechnen diesen von millisekunden runter bis auf Tage oder Wochen (1000/60/60/24/7)
    Dann müssen wir das Zeitformat auf UTC stellen, wegen der Zeitumstellung in
der GMT Zetiform.*/
       //Feiertag bestimmt und in einer funktion verglichen ob das aktuelle datum
mit dem Feiertag übereinstimmt
    buildInfoText();
}
function setDate(newDate){
    selectedDate = newDate;
    buildCalender();
}
    document.getElementById('back_m').onclick = function(){
        setDate
        (new Date(selectedDate.getFullYear(),
            (selectedDate.getMonth()-1),selectedDate.getDate())
        )
    }
```

Könnte auch so aussehen:

```
    /*
      'Kalenderwoche ausrechnen'um die Kalenderwoche zu ermitterl,
      müssen wir den 1.1.xxxx definieren und mit einer schleife
      abfragen, bis wir den ersten Montag im Jahre gefunden haben.
      Das entspricht dann KW1.

      In einer weiteren berechnung ermitterln wir den wert von Kw1
      bis heute und rechnen diesen von millisekunden runter bis auf
      Tage oder Wochen (1000/60/60/24/7). Dann müssen wir das
      Zeitformat auf UTC stellen, wegen der Zeitumstellung in der
      GMT Zetiform.

      Feiertag bestimmt und in einer funktion verglichen ob das
```

```
        aktuelle datum mit dem Feiertag übereinstimmt
    */

    buildInfoText();
}

function setDate(newDate) {
    selectedDate = newDate;
    buildCalender();
}
    document.getElementById('back_m').onclick = function() {
        setDate(
          new Date(
            selectedDate.getFullYear(),
            selectedDate.getMonth()-1,
            selectedDate.getDate()
          );
        )
    }
```

An der Stelle ist aber noch was im Argen. Du siehst selbst, dass die Funktion `setDate` endet und dann haben wir einen Sprung, weil wir eigentlich wieder in der root-Funktion sind, also quasi in der main. Das ist echt unschön. Das übersieht jeder, der den Code zum ersten mal liest. Die ganzen `document.getElementById` gehören nach oben, wo auch der Rest ist, der immer läuft.

Insgesamt kannst Du versuchen, den Code mehr zu strukturieren, also zum Beispiel die `.innerHtml`-Anweisungen in einem Block abzuhandeln.

Die Kommentare auch mehr zum Strukturieren verwenden. Du kannst auch mehr Teilfunktionen verwenden, um ebenfalls noch besser zu gliedern.

Ansonsten passt natürlich die Funktionalität und auch die Einrückungen sind absolut konsistent. Auch der Einsatz der englischen Sprache ist recht durchgängig und das finde ich gut.

Bewertung

Für mich ist das eine befriedigende bis gute Leistung, also verbuche ich sie als 2,5.