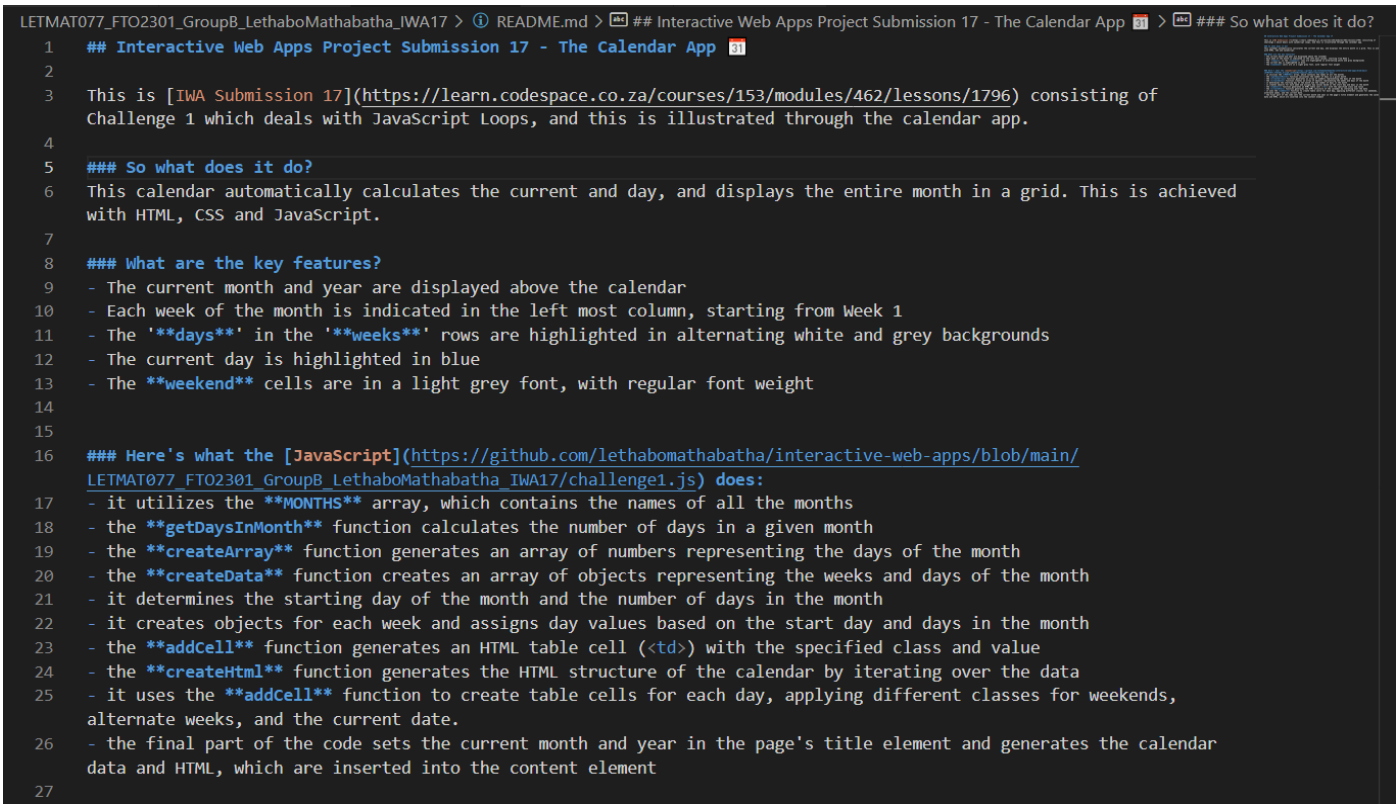
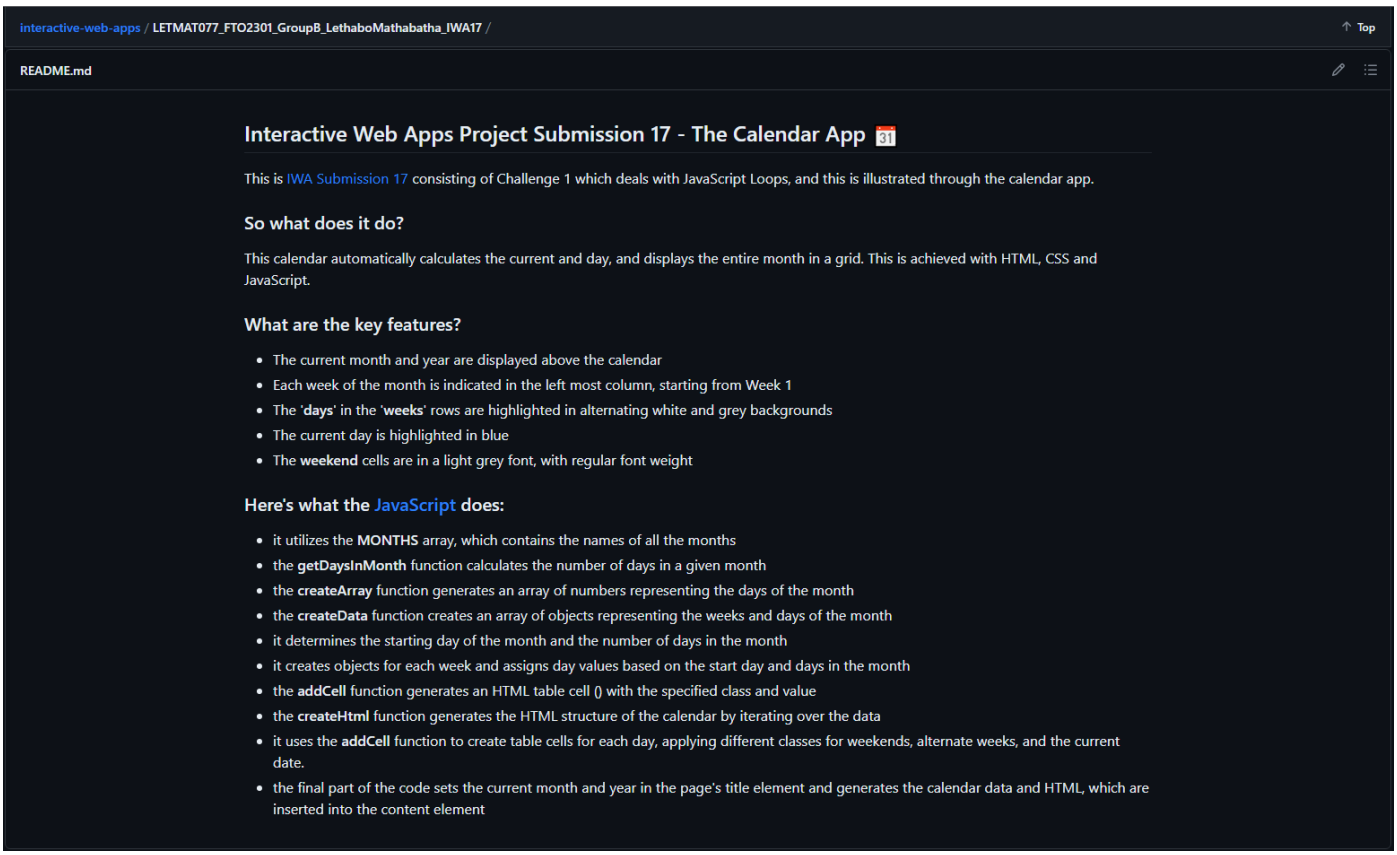
DWA_03.4 Knowledge Check_DWA3.1

---

1. Please show how you applied a Markdown File to a piece of your code.



```
LETMAT077_FTO2301_GroupB_LethaboMathabatha_IWA17 > README.md > ## Interactive Web Apps Project Submission 17 - The Calendar App > ### So what does it do?
## Interactive Web Apps Project Submission 17 - The Calendar App

This is [IWA Submission 17](https://learn.codespace.co.za/courses/153/modules/462/lessons/1796) consisting of Challenge 1 which deals with JavaScript Loops, and this is illustrated through the calendar app.

### So what does it do?
This calendar automatically calculates the current and day, and displays the entire month in a grid. This is achieved with HTML, CSS and JavaScript.

### What are the key features?
- The current month and year are displayed above the calendar
- Each week of the month is indicated in the left most column, starting from Week 1
- The '**days**' in the '**weeks**' rows are highlighted in alternating white and grey backgrounds
- The current day is highlighted in blue
- The **weekend** cells are in a light grey font, with regular font weight


### Here's what the [JavaScript](https://github.com/lethabomathabatha/interactive-web-apps/blob/main/LETMAT077_FTO2301_GroupB_LethaboMathabatha_IWA17/challenge1.js) does:
- it utilizes the **MONTHS** array, which contains the names of all the months
- the **getDaysInMonth** function calculates the number of days in a given month
- the **createArray** function generates an array of numbers representing the days of the month
- the **createData** function creates an array of objects representing the weeks and days of the month
- it determines the starting day of the month and the number of days in the month
- it creates objects for each week and assigns day values based on the start day and days in the month
- the **addCell** function generates an HTML table cell (<td>) with the specified class and value
- the **createHtml** function generates the HTML structure of the calendar by iterating over the data
- it uses the **addCell** function to create table cells for each day, applying different classes for weekends, alternate weeks, and the current date.
- the final part of the code sets the current month and year in the page's title element and generates the calendar data and HTML, which are inserted into the content element

```



interactive-web-apps / LETMAT077_FTO2301_GroupB_LethaboMathabatha_IWA17 / ↑ Top

README.md

Interactive Web Apps Project Submission 17 - The Calendar App 📅

This is IWA Submission 17 consisting of Challenge 1 which deals with JavaScript Loops, and this is illustrated through the calendar app.

So what does it do?

This calendar automatically calculates the current and day, and displays the entire month in a grid. This is achieved with HTML, CSS and JavaScript.

What are the key features?

- The current month and year are displayed above the calendar
- Each week of the month is indicated in the left most column, starting from Week 1
- The '**days**' in the '**weeks**' rows are highlighted in alternating white and grey backgrounds
- The current day is highlighted in blue
- The **weekend** cells are in a light grey font, with regular font weight

Here's what the JavaScript does:

- it utilizes the **MONTHS** array, which contains the names of all the months
- the **getDaysInMonth** function calculates the number of days in a given month
- the **createArray** function generates an array of numbers representing the days of the month
- the **createData** function creates an array of objects representing the weeks and days of the month
- it determines the starting day of the month and the number of days in the month
- it creates objects for each week and assigns day values based on the start day and days in the month
- the **addCell** function generates an HTML table cell () with the specified class and value
- the **createHtml** function generates the HTML structure of the calendar by iterating over the data
- it uses the **addCell** function to create table cells for each day, applying different classes for weekends, alternate weeks, and the current date.
- the final part of the code sets the current month and year in the page's title element and generates the calendar data and HTML, which are inserted into the content element

2. Please show how you applied JSDoc Comments to a piece of your code.

interactive-web-apps / LETMAT077_FTO2301_GroupB_LethaboMathabatha_IWA17 / challenge1.js ↑ Top

Code Blame 139 lines (112 loc) · 3.3 KB Raw

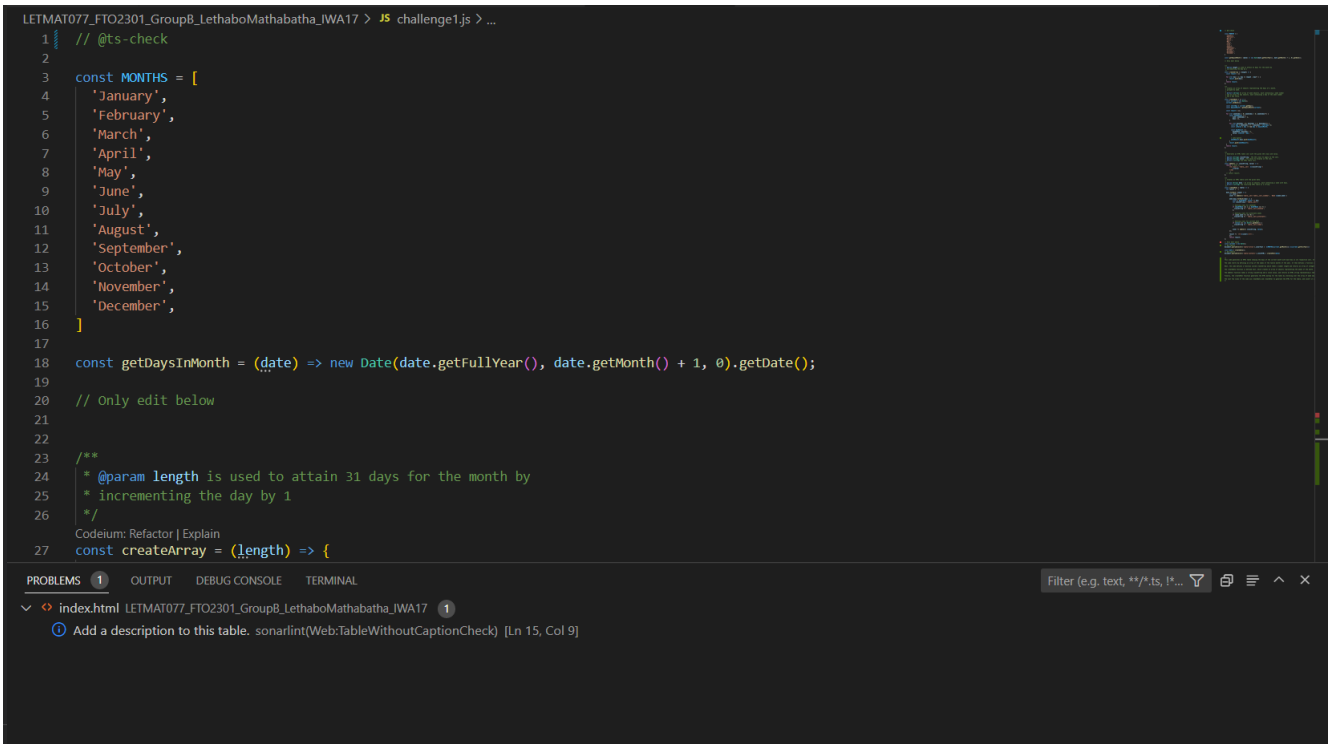
```
23  /**
24   * @param length is used to attain 31 days for the month by
25   * incrementing the day by 1
26   */
27  const createArray = (length) => {
28    const result = []
29
30    for (let day = 1; day <= length ; day++ ) {
31        result.push(day);
32    }
33    return result;
34  };
35
36  /**
37   * Creates an array of objects representing the days of a month,
38   * grouped by week.
39   *
40   * @return {Array} An array of week objects, each containing a week number
41   * and an array of day objects, each containing a day of the week number
42   * and a day value.
43   */
44  const createData = () => {
45    const current = new Date();
46    current.setDate(1);
47
48    const startDay = current.getDay();
49    const daysInMonth = getDaysInMonth(current);
50
51    const result = [];
52
53    for (let weekIndex = 0; weekIndex < 6; weekIndex++) {
54        const weekObject = {
55            week: weekIndex + 1,
56            days: [],
57        }
58
59        for (let dayIndex = 0; dayIndex < 7; dayIndex++) {
60          const day = weekIndex * 7 + dayIndex - startDay +1;
61          const isValid = day > 0 && day <= daysInMonth;
62
63          const dayObject = {
64            dayOfWeek: dayIndex + 1,
65            value: isValid ? day : '',
66          }
67
68          weekObject.days.push(dayObject);
69        }
70        result.push(weekObject);
71    }
72    return result;
73  };
```

```
74
75
76      /**
77       * Generates an HTML table cell with the given CSS class and value.
78       *
79       * @param {string} classString - The CSS class to apply to the cell.
80       * @param {string} value - The value to display in the cell.
81       * @return {string} The HTML table cell.
82       */
83      const addCell = ( classString, value) => {
84        return /* html */ `
85            <td class = 'table__cell  ${classString}'>
86                ${value}
87            </td>
88            `;
89        // return result;
90      };
91
92      /**
93       * Creates an HTML table with the given data.
94       *
95       * @param {Array} data - An array of objects, each containing a week with days.
96       * @return {string} The resulting HTML table as a string.
97       */
98      const createHtml = (data) => {
99        let result = '';
100
101       data.forEach ((week) => {
102           let inner = ''
103           inner += addCell('table__cell table__cell_sidebar', `Week ${week.week}`)
104
```

```
105           week.days.forEach((day) => {
106               const { dayOfWeek, value } = day;
107               let classString = 'table__cell';
108
109               // Setting class for weekends
110               if (dayOfWeek === 1 || dayOfWeek === 7) {
111                 classString += ' table__cell_weekend';
112               };
113
114               // Setting class for alternate weeks
115               if (week.week % 2 === 0) {
116                 classString += ' table__cell_alternate';
117               }
118
119               // Setting class for current date
120               if (value === new Date().getDate()) {
121                 classString += ' table__cell_today';
122               }
123
124               inner += addCell( classString, value);
125           });
126
127           result += `<tr>${inner}</tr>`;
128           });
129           return result;
130       };
131
132       // Only edit above
```

---

3. Please show how you applied the @ts-check annotation to a piece of your code.



---

4. As a BONUS, please show how you applied any other concept covered in the 'Documentation' module.

*****
