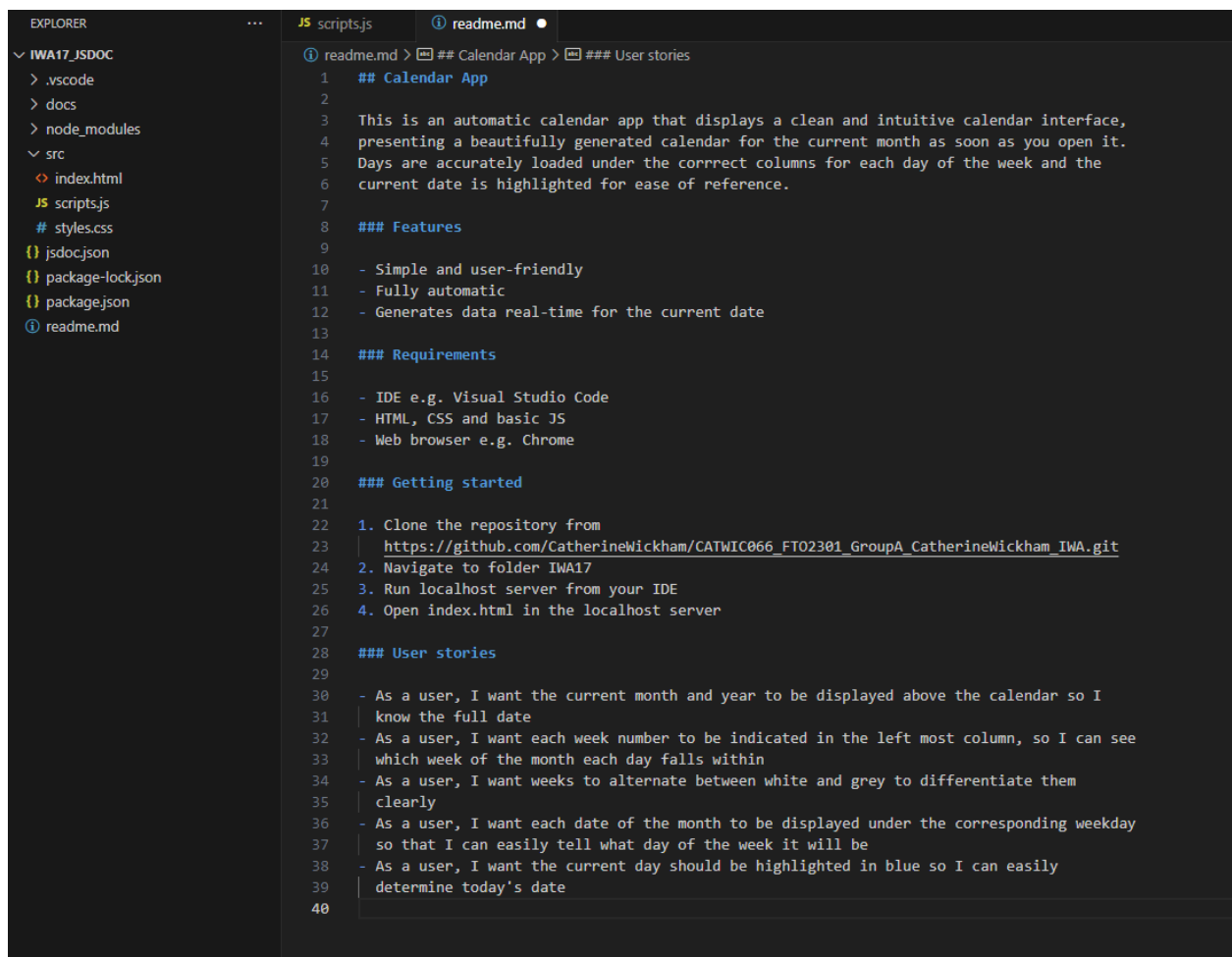


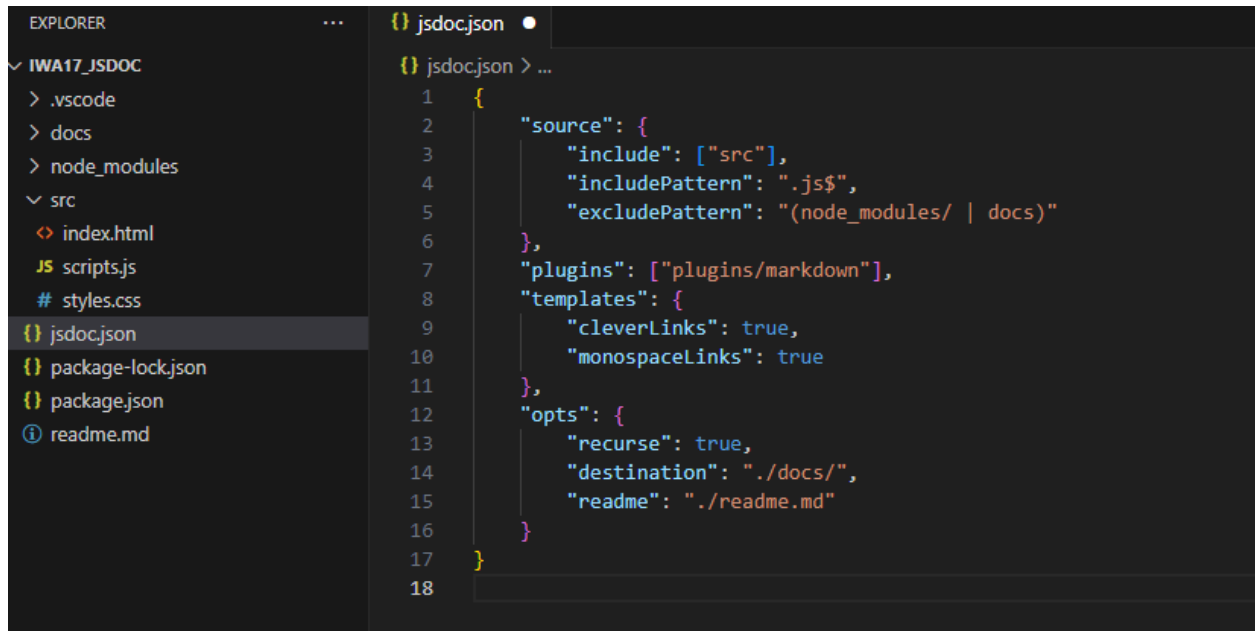
DWA_03.4 Knowledge Check_DWA3.1

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

A readme.md file was added to the root folder of the project to give details about the app and how to operate it. I also linked the readme to my JSDoc website by including "readme": "./readme.md" under the "opts" in my json config file so that it also displays these details on the homepage.



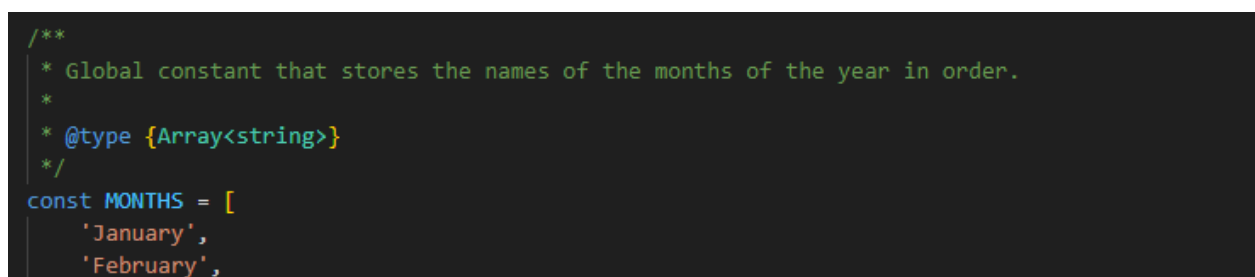
```
1  ## Calendar App
2
3  This is an automatic calendar app that displays a clean and intuitive calendar interface,
4  presenting a beautifully generated calendar for the current month as soon as you open it.
5  Days are accurately loaded under the correct columns for each day of the week and the
6  current date is highlighted for ease of reference.
7
8  ### Features
9
10 - Simple and user-friendly
11 - Fully automatic
12 - Generates data real-time for the current date
13
14 ### Requirements
15
16 - IDE e.g. Visual Studio Code
17 - HTML, CSS and basic JS
18 - Web browser e.g. Chrome
19
20 ### Getting started
21
22 1. Clone the repository from
23 | https://github.com/CatherineWickham/CATWIC066\_FT02301\_GroupA\_CatherineWickham\_IWA.git
24 2. Navigate to folder IWA17
25 3. Run localhost server from your IDE
26 4. Open index.html in the localhost server
27
28 ### User stories
29
30 - As a user, I want the current month and year to be displayed above the calendar so I
31 | know the full date
32 - As a user, I want each week number to be indicated in the left most column, so I can see
33 | which week of the month each day falls within
34 - As a user, I want weeks to alternate between white and grey to differentiate them
35 | clearly
36 - As a user, I want each date of the month to be displayed under the corresponding weekday
37 | so that I can easily tell what day of the week it will be
38 - As a user, I want the current day should be highlighted in blue so I can easily
39 | determine today's date
40
```



```
{
  "source": {
    "include": ["src"],
    "includePattern": ".js$",
    "excludePattern": "(node_modules/ | docs)"
  },
  "plugins": ["plugins/markdown"],
  "templates": {
    "cleverLinks": true,
    "monospaceLinks": true
  },
  "opts": {
    "recurse": true,
    "destination": "./docs/",
    "readme": "./readme.md"
  }
}
```

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

I applied JSDoc Comments with @type keywords to all variables. For functions, I applied @param and @returns keywords wherever possible, with specified types and descriptions for each keyword. Detailed descriptions of what each piece of code does were included in the main comment bodies for clarity.



```
/**
 * Global constant that stores the names of the months of the year in order.
 *
 * @type {Array<string>}
 */
const MONTHS = [
  'January',
  'February',
```

```

/**
 * Function that determines the number of days that are in the month of the input date.
 *
 * @type {Function}
 * @param {Date} date - Input date
 * @returns {number} - Number of days in the month of the input date
 */
const getDaysInMonth = (date) => new Date(date.getFullYear(), date.getMonth() + 1, 0).getDate()

/**
 * Function that generates an array of a specified length which has only placeholder null
 * elements at each position in the array. This is used as a template to loop over and
 * fill in values.
 *
 * @type {Function}
 * @param {number} length - Target length of the array to be created
 * @returns {Array<null>} - Resulting array of target length filled with null elements
 */
const createArray = (length) => {
  let result = []
  for ( let i = 0; i < length; i++) {
    result.push(null)
  }
  return result
}

```

```

/**
 * Function that generates data arranged in nested arrays of objects that will be used to
 * populate the calendar table with day number values.
 *
 * The current date at the time of running the app is determined (`current`) and the day
 * of the week on which the first day of the current month occurred (`startDay`) is
 * calculated from it. The number of days in the current month (`daysInMonth`) is also
 * calculated using {@link getDaysInMonth}.
 *
 * The `daysArray` variable is then created to store the values of the day numbers that
 * will be added for each day of the month. Empty strings are used where whitespace is
 * required instead of a day number (i.e. for days in the calendar that fall outside the
 * current month). The placement of the values/whitespace within the array is based on the
 * `startDay` and `daysInMonth` values.
 *
 * An outer array called `weeks` is created based on a template generated by
 * {@link createArray}. Each element contains an object corresponding to each week of the
 * month which contains the week number and an array called `days`. The template for
 * `days` is also generated using {@link createArray}. Each element of the `days` array is
 * also an object, containing the number of that day within the current week (`dayOfWeek`)
 * and the `value` of that day number (i.e. the `dayOfMonth`) which is assigned for that
 * day from `daysArray` according to the `daysArrayIndex` number.
 *
 * @type {Function}
 * @returns {Weeks} - An array of objects containing week numbers and arrays of day data
 * for each week of the current month. The day arrays contain objects for each day, with
 * day numbers and values for which day of the month the day corresponds to.
 */
const createData = () => {
  const current = new Date
  current.setDate(1)
  const startDay = current.getDay()
  const daysInMonth = getDaysInMonth(current)

  let daysArray = []
  for ( let i = 1; i <= 42; i++) {
    if (i <= startDay || i > daysInMonth + startDay) {
      daysArray.push("")
    } else {
      daysArray.push(i - startDay)
    }
  }

  let weeks = createArray(6)
  let dayOfMonth = null
  let daysArrayIndex = 0

```

```

/**
 * Generates a template string containing HTML for a single new cell of the table in which
 * the calendar is displayed. Appropriate CSS classes are included to format the cell
 * correctly. The new cell is appended to the existing contents of the table as a td
 * element.
 *
 * @type {Function}
 * @param {string | HTMLElement} existing - contains the current state of the HTML
 * template string
 * @param {string} classString - specifies the CSS classes to be applied to the new cell
 * @param {number} value - the number corresponding to the day of the month to be entered
 * into the new cell
 * @returns {string} - template string containing the HTML of the new cell
 */
const addCell = (existing, classString, value) => {
  const result = /* html */ `
    ${existing}
    <td class="${classString}">
      ${value}
    </td>
  `
  return result
}

```

```

/**
 * Generates a template string containing HTML for the entire calendar table, using the
 * data provided. This will later be applied to the DOM by setting the innerHTML of the
 * content section of the app to the value of the final template string.
 *
 * For each week of the weeks array, the cells of the sidebar containing the week numbers
 * are added using {@link addCell}.
 *
 * For each day within that week, checks are performed to see if the day meets any special
 * conditions that require specific formatting (is the current day, or is a weekend day, or
 * falls within an alternate week) and the classString is updated accordingly.
 *
 * The value of the day is set according to the corresponding value in the supplied data
 *
 * The HTML cell for that day is then created using {@link addCell} and added inside a
 * <tr> element in the `result` template string
 *
 * At the end of each week, the results for that week contained in the `result` template
 * string as a single tr element are added to the `combinedResults` template string,
 * resulting in a sequence of tr elements - one for each week
 *
 * @type {Function}
 * @param {Weeks} data - An array containing data for each week of the current month
 * @returns {string} - template string containing the HTML of the entire calendar table as
 * a sequence of tr elements from each week
 */
const createHtml = (data) => {
  let result = ''
  let combinedResults = ''

```

```

/**
 * The current date at the time of running the app, used to set the month name and year
 * displayed above the calendar table
 *
 * @type {Date}
 */
const currentDay = new Date()
// @ts-ignore
document.querySelector('[data-title]').innerText = `${MONTHS[currentDay.getMonth()]} ${currentDay.getFullYear()}`
// should be stored as variable to document properly

/**
 * Contains data generated for the current month using {@link createData}, which is then
 * used to create the calendar table using {@link createHtml}
 *
 * @type {Weeks}
 */
const data = createData()
// @ts-ignore
document.querySelector('[data-content]').innerHTML = createHtml(data)
// should be stored as variable to document properly

```

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

@ts-check was applied at the top of the scripts file to ensure that static type checking was applied. This means that the code is checked for inconsistencies in the typing of data compared to the expected types declared in the JSDoc comments. I did have to apply @ts-ignore to the last few lines of code, as they were giving an error that the result could be null, but as these lines were not properly stored in variables in the original code, they couldn't be documented correctly

```

JS scripts.js > ...
// @ts-check

```

```

/**
 * Global constant that stores the names of the months of the year in order.
 *

```

```

const currentDay = new Date()
// @ts-ignore
document.querySelector('[data-title]').innerText = `${MONTHS[currentDay.getMonth()]} ${currentDay.getFullYear()}`
// should be stored as variable to document properly

```

```

const data = createData()
// @ts-ignore
document.querySelector('[data-content]').innerHTML = createHtml(data)
// should be stored as variable to document properly

```

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

I included the use of custom types using @typedef for describing the complex data structure comprising nested sets of objects within arrays that were generated to store the calendar data. I set up 4 custom types for the following: the object elements of the days array, the days array itself, the object elements of the weeks array and also the weeks array itself.

```
/**
 * An element of an array containing data for a single day of the week, nested within an
 * array containing data for a single week of the current month
 * @typedef {Object} DaysElement
 * @property {number} dayOfWeek - the number of this day within the current week
 * @property {number} value - contains the value of the day of the month assigned to this
 * day
 */

/**
 * An array containing data for each day within a particular week of the current month
 * @typedef {Array<DaysElement>} Days
 */

/**
 * An element of an array containing data for a single week of the current month
 * @typedef {Object} WeeksElement
 * @property {number} week - number of this week within the current month
 * @property {Days} days - an array containing data for each day of this week
 */

/**
 * An array containing data for each week of the current month
 * @typedef {Array<WeeksElement>} Weeks
 */
```

I also set up a JSDoc website as per the JSDoc Crash course:

Home

Calendar App

This is an automatic calendar app that displays a clean and intuitive calendar interface, presenting a beautifully generated calendar for the current month as soon as you open it. Days are accurately loaded under the correct columns for each day of the week and the current date is highlighted for ease of reference.

Features

- Simple and user-friendly
- Fully automatic
- Generates data real-time for the current date

Requirements

- IDE e.g. Visual Studio Code
- HTML, CSS and basic JS
- Web browser e.g. Chrome

Getting started

1. Clone the repository from https://github.com/CatherineWickham/CATWIC066_FTO2301_GroupA_CatherineWickham_IWA.git
2. Navigate to folder IWA17
3. Run localhost server from your IDE
4. Open index.html in the localhost server

User stories

- As a user, I want the current month and year to be displayed above the calendar so I know the full date
- As a user, I want each week number to be indicated in the left most column, so I can see which week of the month each day falls within
- As a user, I want weeks to alternate between white and grey to differentiate them clearly
- As a user, I want each date of the month to be displayed under the corresponding weekday so that I can easily tell what day of the week it will be
- As a user, I want the current day should be highlighted in blue so I can easily determine today's date

Documentation generated by JSDoc 4.0.2 on Fri May 26 2023 09:54:57 GMT+0200 (South Africa Standard Time)

Home

Global

MONTHS
addCell
createArray
createData
createHtml
currentDay
data
getDaysInMonth