20th February 2024

17:8:7

Export | Import | New Event

1st Thursday No events for this day	**2nd** Friday No events for this day	**3rd** Saturday No events for this day	**4th** Sunday No events for this day	**5th** Monday No events for this day	**6th** Tuesday No events for this day
7th Wednesday No events for this day	**8th** Thursday No events for this day	**9th** Friday No events for this day	**10th** Saturday No events for this day	**11th** Sunday No events for this day	**12th** Monday No events for this day
13th Tuesday Shrove Tuesday	**14th** Wednesday Valentines Day	**15th** Thursday No events for this day	**16th** Friday No events for this day	**17th** Saturday No events for this day	**18th** Sunday No events for this day
19th Monday Meeting Jamie	**20th** Tuesday No events for this day	**21st** Wednesday No events for this day	**22nd** Thursday No events for this day	**23rd** Friday No events for this day	**24th** Saturday No events for this day
25th Sunday No events for this day	**26th** Monday No events for this day	**27th** Tuesday No events for this day	**28th** Wednesday No events for this day	**29th** Thursday No events for this day	

February | 2024

I planned the project into the following steps;

• goals of the site; outline what I want the spa to be and do. Outline certain important features and start to conjure up a design (in my head) based off this. Features like, live date/time, customise event colors, export/import the calendar, use comments

• research into similar products; there are products for everything already, figure out what they're doing, how they look, what they're missing and use that to influence my project.

• design/uml models; come up with designs, starting with paper then digital, making sure they are clearly labeled when I use them later on. Create uml models that help me visualise what I have to do, if a certain task is done more then once turn it into a reusable function - even if its used once, if its appropriate make it a function of its own

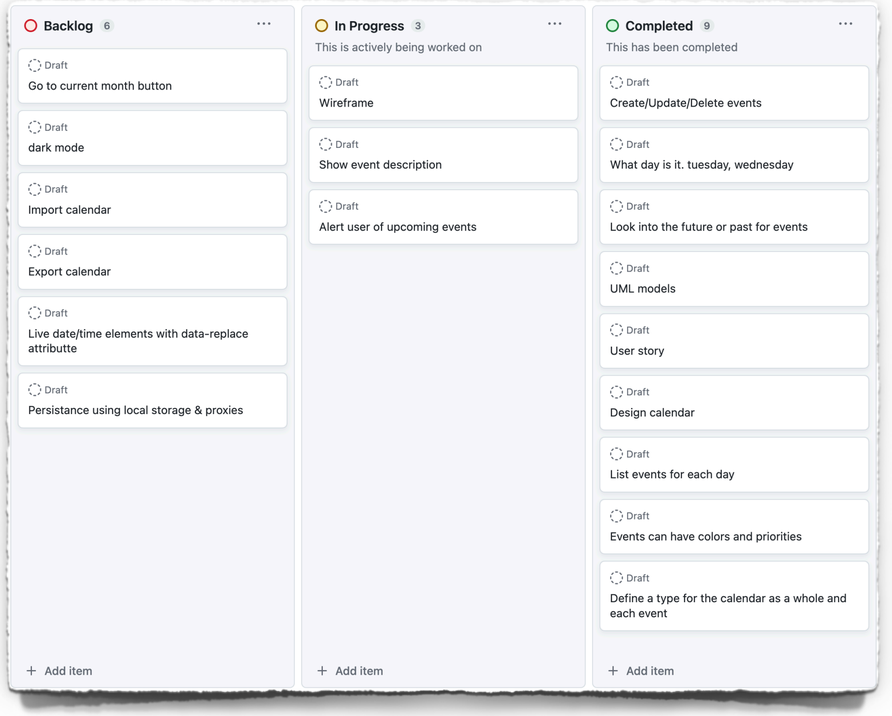
• Code/test it; start actually coding it. Make sure to use comments everywhere (JSDoc), types to reduce errors and componentize it to make it reusable and easily manageable. I used the agile approach as I tend to work in chunks anyway. Label everything clearly, use utility classes in css to reduce size and improve load speed.

• feedback and refactor; show the finished site to people and ask for feedback, implement the feedback. Refactor code that looks slow, chunky or dirty.

Sprints

There were a few “sprints”. I used GitHubs’ projects feature (bottom image) to create a board where I could manage these. My first step was to plan what sprints I would be doing. I could get these from the features I planned (top image) and cherry pick ones worth a sprint. Each sprint I would write something explaining, why im doing it, what im doing and how to tackle it. I’ve put a couple example on the next slides. I would write these in markdown, its simple, easy to learn and quite presentable afterwards.

Feature	Status
Persistence using local storage/proxies	
Day (Tuesday, Monday) on each date in calendar view	
Ability to change selected month in calendar view	
Create, update, delete events	
Go to current month button if current month != selected month	
Dark mode/light mode based on color scheme	
Import calendar JSON	
Export calendar JSON	
Banner for todays events (hides if 0 events toady)	
Events have custom colors and priority	
Live date/time with html attributes	
Define calendar data/calendar event type	



Types

In this meeting we discussed how Kanban boards help the workflow of the task overall and how we can track different things we need to do. You will have 3 columns; backlog, in progress and done. this helps the flow of tasks and overall productivity. it helps track what has priority and get things done in order. whilst i'm not in a team setting when i do it will help the flow of tasks between multiple people. The first thing to do is to figure out the type of a single event and the entire calendar. This is super important as it helps with intellisense but also errors as you know what you need to create an event, update, and helps with type checking. This is as follows;

```
// typedef CalendarEvent
{
    "title": string, // "Event title"
    "allDay": true // has to be true otherwise will have a start time and end time
    "description": string, // "Event description"
    "urgent": boolean, // false
    "color": string, // "#fff000"
    "id": string // "xxxxx-yyyyy-zzzzz"
}

// or

{
    "title": string, // "Event title"
    "startTime":  {
        hour: number, // 12
        minute: number // 0
    },
    "endTime": {
        hour: number, // 15
        minute: number // 30
    },
    "description": string, // "Event description"
    "urgent": boolean, // false
    "color": string, // "#fff000"
    "id": string // "xxxxx-yyyyy-zzzzz"
}
```

```
// demo-calendar.json
{
    "[year]": { // 2023
        "[month]": { // 12
            "[date]": [ // 1
                [Event], [Event], [Event] // Event is a type as described above
            ]
        },
        "[month]": { // 9
            "[date]": [ // 12
                [Event], [Event], [Event] // Event is a type as described above
            ]
        }
    }
}
```

29/11/2023

Types 2

I have been facing trouble with the speed of interaction with the events. It's also a bit complicated for my liking so to combat this i am changing the type of an event and the calendar data as a whole. I'm going to simplify it and make it smaller, this will make it easier to interact with and also easier to understand. The new type is as follows;

```
// typedef CalendarEvent
{
    "title": string, // "Event title"
    "description": string, // "Event description"
    "id": string // "xxxxx-yyyyy-zzzzz"
    "priority": 0 | 1 | 2 | 3, // 1
    "hue": number, // 45
    "begin": string, // "2024-02-21T12:00:00"
    "end": string // "2024-02-21T15:30:00"
}
```

```
// demo-calendar.json
{
    "events": [] // array of CalendarEvent
}
```

21/02/2024

Persistance

One of the last sprints to do. After speaking to people and gaining their feedback it has come to light that everyone wants it to be persistent. This should be fairly obvious but they stated how useful is a calendar if it can't remember what your doing. To implement this i will be using 2 things, proxies and local storage. Local storage is the browsers local store for information. It is a key value pair object (value has to be string). Although you can only store strings you can use JSON parse and stringify to store objects and other data types. The other thing we will use is proxies, these allow for event listeners on objects which is perfect as it keeps the amount of code we need to write to a minimum while doing exactly what we're after. The code has to be simple, readable and understandable. Another big thing is having the code componentised so that it can be reused and if your debugging something it is much easier to figure out where a problem could be and fix it. Using local storage makes sure that the events are saved when the page was closed and when its opened again the events are still there. Persistance is vital to having the calendar be useful practically.

15/02/2024

init()

- startLiveDateTime()
- Check if calendar in local storage
- If exists
  - Calendar data = local storage calendar
- Else
  - Load calendar data from demo file
  - Save calendar data to local storage
- Refresh banner/calendar UI

Click "New Event"

- resetModal("new-event")
- openModal("new-event")
- onSubmit
  - closeModal("new-event")
  - Read data from modal
  - save new event to calendar data
  - update local storage accordingly
  - Refresh banner/calendar UI
- onClose
  - closeModal("new-event")

Click "Delete Event"

- closeModal("update-event")
- Remove event from calendar data
- Update local storage
- Refresh banner/calendar UI

startLiveDateTime()

- updateDateTime()
- setInterval(updateTime, 1000)

updateDateTime()

- Get current date
- Format date into date, year, hour, etc
- Get all elements with [data-replace-with] attribute
- Go through each element
  - replace the value of [data-replace-with] with the according value from formatted time
  - Set the textContent to the replaced value

Click on any event

- loadUpdateEventModal(eventData)
- openModal("update-event")
- onSubmit
  - remove event from calendar data
  - add new event from modal to calendar data
  - update local storage
  - refresh banner/calendar UI
- onClose
  - closeModal("update-event")

Click "Export Calendar"

- JSON.stringify(calenda data)
- downloadFile("calendar.json", stringifiedData)

Click "Import Calendar"

- Open file input dialog
- onSubmit
  - Read file
  - try to parse it
    - if parsed success
      - Save to calendar data
      - update local storage
      - refresh banner/calendar UI
    - Else
      - alert user file not formatted
- onCancel
  - Do nothing

downloadFile(name, contents)

- Create fake anchor tag
- Add download attribute with name param
- Create blob from contents
- Assign blob to href
- click the <a> with JS

The challenges I faced;

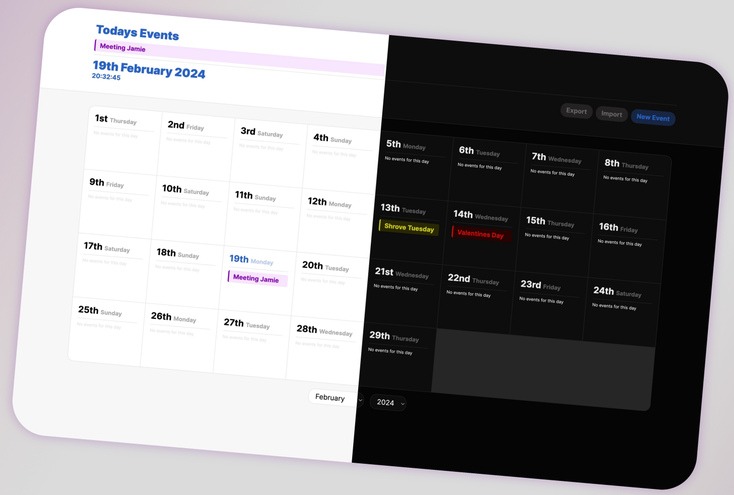
The first was creating the type for events and the calendar overall. It shouldn't have been but because there were so many ways I could've done it I kept second guessing my self and re typing it. I ended up on the first iteration which was a dirty solution but I was happy to have stuck on one. I kept with it for a little while but eventually move to iteration 2 which was simpler and cleaner. This create the issue of going through all the code and whenever an event is read, updated or deleted having to rewrite it to the new type. I removed the key starts so whenever event.starts was accessed would return null. Posed quite an issue and very time consuming.

Another issue which which was very silly was having an event beginning and end. At the start I had an all day boolean and a start/end time/date - but all stored in its own key. At the time I didn't realise how much this was slowing the coding process down, having to format and concatenate every time I needed the date. After going in circles - for some reason - I ended up landing on saving 2 values only; begin and end. Those are the keys and the values are a local date/time all in one. Much simpler, no formatting and cleaner.

A smaller issue was coming up with ways to style the event on the screen. I started with nothing, then styled the text color, then removed the color but added a rounded background, then removed that. Then added a border to one side that was the same color as the text and the background was a lighter color of that (darker for dark mode). Then I faced the issue of how does the user set that color. I remembered in photoshop you can change the hue of a color to change the actual color but not the saturation or lightness, so if the user can set the hue I can control the rest creating a custom palette based off one value. Only thing I need now is for the user to pick any color (native color picker) and I will get the hue from that color

The Last Calendar

Welcome to my innovative calendar where organization meets simplicity. This is the last calendar you will ever need! Prioratizing user-friendly UI, designed to streamline your daily activities. Stay on top of your plans effortlessly and discover a new way to manage your time effectively. Unlock the power of efficient planning today!



Features

The calendar its self is designed with ease in mind. Making sure no matter what your doing it will come easy. Speed is another huge priority, you will not experience any lag. Not only is the calendar saved locally on the browser but you can import & export your own calendar too. Easily see the type of event via colors that you can set too whatever you desire.

How Its Made

The calendar is made entirely using vanilla HTML, CSS & JavaScript. In doing this there is no bloatwear. Only what i use is loaded in the browser therefore it is small and fast. Use the browser tools to take a look at how its made, the code is clean, formatted and organised.

Bobby Mannino

comp-1004-spa.vercel.app

bobby.mannino@students.plymouth.ac.uk