UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS 216 Practical Assignment 2

- Date Issued: **14 March 2022**
- Date Due: **28 March 2022** before **08:00**
- Submission Procedure: **Upload to the web server (`wheatley`) + clickUP**
- This assignment consists of **3 tasks** for a total of **100 marks**.

1 Introduction

During this practical assignment you will be designing and developing a web site that will showcase a News Database similar to what can be seen from most news websites, e.g. `https://www.news24.com/`. Each assignment will build off the other in an attempt to have a fully functional news listing website at the end of all the practicals. **NB: It is important that you do not miss any practicals or you will fall behind.**

After successful completion of this assignment you should be able to create a web page which complies to the HTML5 and JavaScript Standards. The specific web page for this assignment will showcase the following functionality:

- JavaScript animations
- Loading Screen
- Retrieving data from various APIs
- Populating your templates with the retrieved API data
- Implementing the Calendar page and functionality

2 Constraints

1. You must complete this assignment individually.
2. You may ask the Teaching Assistants for help but they will not be able to give you the solutions.
3. You must produce all of the source files yourself; you may not use any tool to generate source files or fragments thereof automatically.
4. Your assignment will be viewed using Brave Web Browser (`https://brave.com/`) so be sure to test your assignment in this browser. Nevertheless, you should take care to follow published standards and make sure that your assignment works in as many browsers as possible.
5. You may utilise any text editor or IDE, upon an OS of your choice, again, as long as you do not make use of any tools to generate your assignment.
6. All written code should contain comments including your name, surname and student number at the top of each file.
7. Your assignment must work on the `wheatley` web server, as you will be marked off there.
8. **You must only use JavaScript for this assignment (no libraries), but you may use JQuery.**

3 Submission Instructions

You are required to upload all your source files (e.g. HTML5 documents, any images, etc.) to the web server (`wheatley`) and clickUP in a compressed (zip) archive. Make sure that you test your submission to the web server thoroughly. All the menu items, links, buttons, *etc.* must work and all your images must load. Make sure that your practical assignment works on the web server before the deadline. No late submissions will be accepted, so make sure you upload in good time. The server will not be accepting any uploads and updates to files from the stipulated deadline time until the end of the marking week (Thursday at 3pm).

The deadline is on Sunday but we will allow you to upload until Monday 8am. After this NO more submissions will be accepted.

Note, `wheatley` is currently available from anywhere. But do not rely that outside access from the UP network will always work as intended. You must therefore make sure that you `ftp` your assignment to the web server. Also make sure that you do this in good time. A snapshot of the web server will be taken just after the submission was due and only files in the snapshot will be marked.

4 Online resources

HTML5 - `https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5`
`http://www.w3schools.com/`

CSS - `http://www.w3.org/Style/Examples/011/firstcss`

Javascript - `https://www.w3schools.com/js/`
`https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics`

REST - `https://en.wikipedia.org/wiki/Representational_state_transfer`

JSON - `https://www.w3schools.com/js/js_json_syntax.asp`

HTTP Methods - `https://www.w3schools.com/tags/ref_httpmethods.asp`

AJAX - `https://www.w3schools.com/js/js_ajax_intro.asp`

Chart.js - `https://www.chartjs.org/`

5 Rubric for marking

News Article listing	
Data population	5
Searchable	5
Filterable	5
Loading Screen	5
Relevant articles on each page	5
DOM manipulation	5
API Calls	
Makes use of multiple APIs	4
AJAX	12
JSON manipulation	4
Calendar	
Layout	6
Buttons and Functionality	10
Entries	4
COVID-19 Page	
Layout	5
Statistics	5
Deductions	5
Graph	10
Validation	
HTML	3
JS	2
Upload	
Does not work on `wheatley`	**-20**
Not uploaded to clickUP	**-100**
Bonus	5
Total	**100**

6 Assignment Instructions

Task 1: Populate from APIs ..... (50 marks)

This task requires you to pull information from **various APIs** and populate all the templates (HTML layouts) you have created in Assignment 1. You should retrieve data from external APIs for at least 20 news articles (and a minimum of 5 per page).

All API calls must be done through JavaScript **AJAX XMLHttpRequest** (`https://www.w3schools.com/js/js_ajax_intro.asp`). You must choose between synchronous and asynchronous calls and provide reasons for your choice.
NB: Your reasoning should be included at the top of your JS file/s as a comment.

You may use, but are not limited to the following APIs:

Google News API - `https://rapidapi.com/newscatcher-api-newscatcher-api-default/api/google-news`

Hacker News - `https://hackernews.api-docs.io/v0/overview/introduction`

NewsAPI.org - `https://newsapi.org/`

TheNews - `https://thenewsapi.com/`

NewsData - `https://newsdata.io/docs`

Disease.sh - `https://disease.sh/docs/#/COVID-19/`

Jokes - `https://jokes.one/api/joke/`

Note : *Don't use APIs that require paid subscriptions. Many APIs allow you a certain number of requests per day on a free tier, just use those requests sparingly (especially on marking days).*

The following functionality needs to be implemented:

- You need to **replace your mock data** from Practical 1 with data that you sourced from at least 2 of the above mentioned APIs (or any others that you find).
- For each API call you need to have a **loading screen/animation** to show that the data is being retrieved. This can be a GIF/APNG/SVG loader for article (it will disappear once the specific news data has loaded) or a loading screen for the entire page (it will disappear once all the data on that page has loaded). Decide which is best. You can get resources from (`https://loading.io/`) and (`https://designmodo.com/`).
- Your website lists news articles. After each news item has been populated from an API, there needs to be a way to be directed to the actual, original article. Preferably it opens in a new tab. You may choose how to present the link to the article.

6.1 Today page

- Implement **search functionality** giving the user the ability to search for a specific article based on the title.
- Implement **filter functionality** using the filters you chose in Practical 1. For example, if you chose to filter by author it should only display articles from that author. **Note:** You should have at least 2 filters.
- Data displayed here should be articles dated to the current date.

6.2 World

- Similar to the today page, but news sourced is sorted from latest to newest over all dates (up to a reasonable point).

6.3 South Africa

- Similar to the World page, but articles that have to do with South Africa.

6.4 COVID-19

- The **statistics data** and **graph** need to be populated. You can use the API from above or an API that you find.
- You need to implement functionality to calculate the deductions on the page. For the deductions, you **may NOT pull** the data from an API, only use API data to calculate the relevant values.
- You need to have a **JS animation** in the form of scrolling text along the screen. Specifically, the page should have an appropriate headline (Such as *COVID-19 Newest Data*) that moves across the page and loops back to where it started. This is similar to how text at the bottom of the screen of a news station scrolls across, except that it keeps looping.
- You need to implement the graph that replaces the placeholder from Prac1. You may use the Chart.js library for this, and display whichever relevant statistic you wish.
  Note: Bonus marks may be given if an HTML Canvas is used to create the graph programmatically.

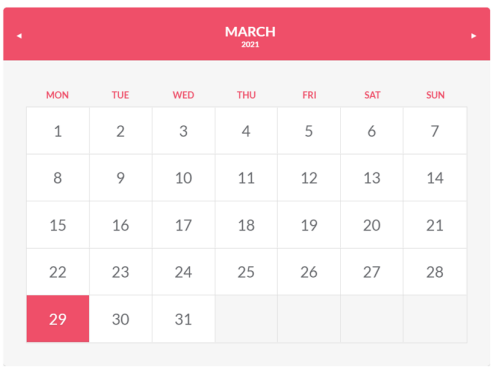
Task 2: Create the "Calendar" Page ..... (20 marks)

You will need to design and implement the Calendar page and functionality whereby you create an entry for each article based on its date. Your Calendar can be in a grid or column format but must have the functionality of a Calendar.
Note: You may NOT use preexisting templates for your calendar and must create it yourself using HTML and CSS. Do not use any generated code or iframes.

The following functionality needs to be implemented:

- Next - button that will advance to the Next month
- Previous - button that will go back to the Previous month
- Today - focus the calendar on today's date
- Month - the name of the month should be shown on the calendar
- Year - the year should be shown on the calendar
- Entry - each entry should have the article's title under the correct release date

NB: Your calendar must be valid. This means that months with 30 days should not be able to show 31 days and the days should match to their actual dates.
For instance, in the screenshot below, March shows 31 days whereas April shows 30, since there are only 30 days in April. The dates displayed under each day also match up with their actual dates (for example, the 29th of March was a Monday this year, so it is shown under the 'MON' column).



Note: You will be awarded marks for the Layout of your calendar so make sure it is well-designed and user friendly.

Task 3: Bonus ..... (5 marks)

Bonus marks will be given for additional functionality and awarded based on the difficulty level. These marks will only be awarded for notable additions (simply adding a background image is not enough).

The following, among others, may earn you extra marks:

- Interactive Calendar with a modal/overlay showing more information about the specific article on hover/click/etc.
- The ability to export the Calendar into a recognised format.
- JS animations to enhance the viewing experience of the Calendar.

- Being able to choose what data the graph displays.
- The graph being animated with JS.