NASA Apollo 11 Mission History Microsite Project Report

1. Introduction

The goal of this semester exam project has been to develop a history learning microsite for the groundbreaking first moon landing mission, in celebration of its 50th anniversary on July 20 2019. Starting from planning the project schedule with the Gantt Chart, defining the functional requirements, and researching the target user group, I designed and developed a responsive, dynamic, and interactive microsite using NASA's Image and Video Library API (accessible at: https://api.nasa.gov/api.html#Images). This report presents and discuss the process of design and development. The project is accessible at:

Github page URL:
https://nancybolstad.github.io/nasa-apollo/
Personal server:
https://nancybolstad.no/Noroff/spring2019/project-exam/apollo/index.html
Github repository:
https://github.com/NancyBolstad/nasa-apollo

2. Planning

In order to be as well prepared as possible before embarking on actually creating the site, I spent the first week on planning out the details of its goals and how these best could be implemented.

As detailed in my earlier planning report, the choice of focusing on the Apollo 11 mission came from the fact that this year marks it's 50th anniversary. As a clearly demarcated event with historical, scientific and educational value, it seemed to fit all the boxes for a topic suited for a microsite, and the timing was perfect. The goal of this project has been to create a functional, dynamic and responsive microsite that

does justice to this event by offering the user a good presentation of the history of the Apollo 11 mission, as well as the ability to explore on his/her own through the search functionality.

To carry out the project, I started from making a project schedule with a Gantt Chart. By doing so, I broke down the project into small tasks and focused on clarifying the dependencies between different tasks.

After conducting research on different API services provided by NASA, I found that the NASA Image and Video Library API would be fitting for the purpose of this project for the following reasons. First, the API includes archival media and descriptions related to all of NASA's space exploration activities and missions over the years. This is very fitting for the purpose of my microsite, which is to present the history of the Apollo 11 mission. Second, the API is mobile friendly with a fast response time. This is essential to ensure fast loading speeds. Third, the API is searchable. Developers can manipulate the API call by including different parameters, so as to narrow down the response by keywords, years or media types. This is practical for my project, as it allows me to get only data related to the Apollo 11 mission from a timeframe of my own choosing.

As for the functional requirements, I decided that the microsite shall contain the following minimum functionality:

1) The microsite must be responsive on different devices.
2) The microsite must have a fast loading speed with quick data retrieval from the remote server.
3) The microsite must be optimized for search engines.
4) The microsite must be accessible. It must comply with the Web Content Accessibility Guidelines (WCAG) and the World Wide Web Consortium (W3C) standards, using semantic HTML tags together with ARIA attributes.

5) The microsite must be easy, obvious and intuitive for the user to use.
6) The microsite must support onsite search.
7) The microsite must contain clearly structured information about the historical event.
8) The microsite must allow for user interaction.

3. Design research: Target audience research

The target audience for this project has been people from the education sector, that is students or teachers, with an emphasis on students from the late high school to early bachelor's level, i.e. of around 17-25 years of age. Teachers would mostly be a target group in the sense that they could use this page as a part of their teaching materials for their students.

To understand the user group's web usage preferences and abilities, I referred to Nielsen Norman Group's user experience research on different age groups. My target group fits well with their categories of young adults or college students. According to their research,:

1) Young adults enjoy interactivity only when the interactivity has a clear purpose and aids the task they are currently working on.
2) While young adults are better readers than early teens, they still prefer scan-friendly content to large chunks of text.
3) Generally speaking, young adults find autoplay of sound or video to be disruptive and therefore dislike such functionality.
4) While they are more patient than kids and young teens, young adults still prefer quick answers.
5) Young adults know how to use and frequently makes use of search functionality.

6) Young adults are confident in their own navigation abilities and site usage skills. When something goes wrong, they are quick to assume that it's the fault of the site and not their own.

With these web usage characteristics of young adults in mind, I developed several strategies for the design of the microsite: 1) the microsite's interactivity should be as intuitive and predictable as possible, since young adults prefer interactivity with a clear purpose; 2) the site should avoid too cluttered and too large chunks of text and the layout should be scan-friendly; 3) video autoplay should be avoided; 4) the information should be presented in a clean, hierarchical, and logical manner so as to accommodate impatience; 5) the microsite should have a good search functionality; 6) the site should have good information for the user if something works out different than one would expect, for instance if data fails to be retrieved from the API, so that the user knows what has happened or went wrong.

4. Graphic design: design principles, typography, color

The graphic design for this microsite follows design principles of contrast, repetition, hierarchy, pattern, emphasis and the usage of white space. Together with the chosen font and color scheme, the visuals of the microsite are aesthetically pleasing, and create a comfortable reading and browsing experience. This section explains how this has been done.

In the color scheme, I used dark cobalt blue (#0054B0) as my brand color, similar to the color of the NASA logo. This is the color that appears on my microsite logo, call-to-action buttons, section headings, and timeline borders. The reasons for using this brand color is that it fits with the space/science theme, as this is a common connotation of deep blue colors. The brand color has been used to emphasize site content such as call-to-action elements and headings. Different tones of this brand color has been applied to create hover effects on buttons and other clickable elements.

Aside from the primary color, I also used a bright color scheme for supporting colors. In my original prototype, I chose to use a dark background. However, because reading text is an important part of the microsite and lighter backgrounds in general are better suited for text readability, I decided to change to a brighter theme.

As with my last semester project, I have made use of sans serif typeface web font "Roboto". This web font is Google's most popular for website development, due to its readability, accessibility, scalability and zoomability. I chose to host the font on my own server in the WOFF2 format in order to ensure fast loading speed and to guard against the potential risks of relying on a third party service.

Together with my color palette and web font, the graphical design of this microsite complies with the following design principles:

First, the design has a good color contrast. Different supporting colors have also been used to separate different sections on each page. For instance on the "Milestones" page, instead of displaying the whole timeline within a single section with only one background color, the line has been separated into meaningful chunks ("Pre-Launch", "Launching" and "Landing") which alternate the background color. Furthermore, I used black font color against a bright background, which makes the content easier to read. Most important, the color contrast on the page has been validated using the *WAVE Web Accessibility Tool*, which shows that the microsite meets the WCAG 2.0 at the conformance level AAA.

Second, the design has a clear hierarchy. Important content, such as leading section titles have been colored in the brand blue color to visually emphasize its importance. I also applied different font-sizes according different types of text. Different font weight has been used to mark emphasis.

Third, the design has a good use of white space. Since the website is rich in information, proper usage of white space can help to avoid overwhelming the user

Fourth, considering the fact that young adults prefer content that is easy to scan and does not like too much cluttered text at one spot, the layout of most of the pages on my microsite follows a Z-pattern. For instance, on the Astronauts and Rockets page, I used the CSS Flex Direction properties to reverse the order of image and text for each row of information, creating a visual Z-pattern. Additionally, for the individual articles I have applied a F-shaped layout. For instance, on the timeline under the Milestones page, each paragraph of text ending with a "View details" button creates such a F-shape, which corresponds to our natural eyes movement when reading content online.

Fifth, repetition has been utilized to create a unified feeling. Throughout the microsite, same colors and same font styles have been applied. I also repeated the same components throughout the microsite. For instance, the clickable cards on the Home page and the Milestones page has the same appearance as their purpose is similar: navigate the user to a different section of the site. Furthermore, on the search results page all of the search result cards are presented with the same size, looks and hover effects. In addition to creating a unified feeling, the usage of repetition also creates predictability for the user.

5.HTML/CSS: Semantics, structure

To create the layout for my content I used semantic HTML5 tags such as <headers>, <main> and <section>. Additionally, I used non-semantic HTML elements such as and for CSS styling purposes. These semantic tags will inform the browser about the structure of the web page content, whereas the non-semantic tags will inform the browser how to display the style of the content. For instance, I used the semantic <section> tag to structure the timeline section and the non-semantic tag for styling the dates in the timeline.

In addition, in order to facilitate web accessibility and to make it easy for search engines to grab the main content of my microsite, I made use of all heading tags from <h1> to <h6> in a hierarchical manner without skipping any heading levels. My page structure has been validated using the WAVE Web Accessibility Tool with no errors found.

6. SEO/Content Strategy/WCAG

For SEO purposes, I added meta tags to the head of each HTML file. The content of the meta description is: "We provide multimedia-rich searchable information about milestones, people and space technology of the Apollo 11 mission, celebrating its 50th anniversary." It was formulated after having first used Google Trends to identify queries and topics related to the Apollo 11 mission. Being descriptive, general and straight to the point about what the website offers, I believe the end result is quite successful. Furthermore, the length of the meta description was kept within 160 characters so that the whole description would be displayed in a Google search result. Lastly, I added a freely available Google Analytics script to my header to generate detailed statistics about visitors and monitor web traffic of my microsite. This could be useful after the website has been launched for purposes of user group research, marketing, etc.

As for the content strategy, to effectively present the multifaceted and complex history of events surrounding the Apollo 11 mission, I chose to rely on popular search terms to help create page categories. Examples of such search terms are: Apollo 11 timeline, Apollo 11 astronauts, Apollo 11 rockets. Based on these popular search terms, I was able to limit and structure the information architecture on my microsite. Of course, the majority of the content is dynamically retrieved from the NASA API, relying on their rich historical archives.

In terms of web accessibility, as stated above, both the site structure as well as the color contrasts meet the WCAG requirements. I used ARIA labels for links and form elements and gave ALT attributes for images. Furthermore, the lead video on the home page has taken accessibility into concern, as it includes a caption and audio description about the video content. For actually captioning and creating a text-to-speech audio description of the video, special software would be required. Due to time constraints, only placeholder versions have been included in this project. They are included to show that I have given the accessibility of the video concern.

7. Interface Design

Taking into account the personas I created during the planning stage of this project, certain aspects of the interface design has been tailored to their preferences.

Let's use the example of the persona Karl, to describe the onsite search functionality. First, Karl would certainly find this function handy when looking for articles related to specific topics/aspects of the Apollo 11 mission. I placed the search icon on the rightmost part of the top navigation bar, which is the most common placement for search functionality. Thus, Karl will quickly be able to locate the search area. After having clicked the icon, Karl will be presented with a search overlay screen which gives a good concentration of the search related functionality. Karl now has the option of typing in his own search term, and the field for doing so has a handy example of what he might search for. Or, if he is on the desktop version of the site, he may choose one of the presented popular search keywords through a single click.

The search form is easy to use, case insensitive and can be submitted either through pressing the Enter key or through clicking the Search button. Having made a search (either by typing his own search or by clicking one of the suggested popular searches), he will be navigated directly to the search results page.

The search results page will give useful information about how many results were found. For instance, for “Apollo Legacy”, 10 search results were found. Do note, however, that due to a default in the API - which may not be changed - no more than a maximum of 100 results will be shown.

When the user hovers over a search result, the hover effect will be triggered, zooming in on the image, changing the font color and highlighting a view details options. When clicking a result, the user will be navigated into a page with details for that specific article. Furthermore, the search results page also has a search form, so that the user can make a new search immediately on the same page, without having to navigate to the search section again. Aside from this, while the search results are loading, a loading indicator will display as a spinner, so as to indicate to the user that a search is currently ongoing. Last but not least, an intuitive message will be presented to Karl if the API call fails.

Another interface design that I put a lot of effort on is to create a good navigation. As for the top navigation bar, I made it stick to the top while the user scrolls the page. The navigation bar is also responsive. For users browsing the site through mobile devices, a clickable, drop-down toggle hamburger menu will appear on the leftmost part of the top navigation bar, whereas the centre of the navigation bar contains the site logo and the rightmost part of the bar is the search icon. However, on the desktop version the site logo is on the leftmost part of the top navigation bar, whereas all the other navigation links are on the left side. In addition to this, I placed a top progress indicator right below the top navigation bar, which will show the users their current position on the current page while he/she is scrolling.

Furthermore, a sort of subtle navigation is provided inside different pages. For instance, the heading – “Explore More About The Apollo Moon Landing” – will be displayed on the page the user is taken to, after he/she has successfully submitted a contact form. This gives the user some options for further navigation. Another

example is on the details page, which based on the current article will show other relevant articles under the heading “You May Also Be Interested In”. A different kind of subtle navigation is provided on the home page. Here I set the page hero fixed to a height of 95 percent of the screen, which means there will always be a narrow white field at the bottom making it clear to the user that he/she may scroll down to get the remainder of the page. The big contrast between the dark hero image and the white border below emphasises this strongly. In addition, there is a floating astronaut image on the centre, bottom of the screen, of which only half will be showed before scrolling. This is an element I chose to include in order to further reinforce the point that the page has further content below. These implicit navigation features are not only functional and intuitive but also visually attractive and thematically fitting.

The URL is also an important part of page navigation, since it provides specific information about the user’s current location. Therefore, all the URLs on the microsite are composed of readable and descriptive parameters and file names, such as “milestones.html”, “search-results.html”, “form-submitted.html”, etc. Similarly, after the user has submitted the search field input, the search results page URL will update with the searched term as a query at the end of the URL endpoint.

Aside from the above mentioned points, my interface design also takes into account the principles of affordances and persuasion. Affordance is implemented on my microsite through cues such as: the search and hamburger icons that tell the user what function to expect from these; the view more buttons which contain not only text but also animated right arrows, which will move upon mouse hover to emphasize its clickability.

As for the persuasion principle, heading titles have been carefully crafted in order to create a credible feeling. For instance, on the home page a section called “Editor’s top picks” is included to give the user a highlight that simultaneously urges him/her to

continue browsing these specially selected articles. Overall, the general consistency throughout the microsite also contributes to creating a credible atmosphere.

8. Javascript:

The user experience and the site performance has been significantly improved by using Javascript. This section explains how Javascript has been used in this project.

First, I included the <script> inside the <head> tag of the HTML file with a defer attribute. This is to ensure fast page loading when using external scripts, since the defer attribute will ensure that the script is loaded asynchronously and that its execution will not take place until after the HTML parsing has been completed.

Second, I used the async/await functions together with the fetch API to communicate with the NASA API server asynchronously, and together with the try/catch block, so as to ensure that data fetching errors can be easily handled after being caught. By using the async/await functions, I also avoided deeply nested callback functions, since the script will wait to execute the next function until the promise has been resolved. Additionally, I also used template literals for URL string concatenations, a handy feature from JavaScript ES6+. Furthermore, I used the encodeURIComponent function, to encode query strings before passing them into the url string. This is to ensure that these query strings are always interpreted correctly, for instance if the user's search term contains whitespaces between words.

I also created a sophisticated data fetching function in the details.js file to dynamically retrieve data based on the related keywords. While the details page is being loaded, the page will first make an API call based on the ID-parameter of the current URL query string. This will fetch the result from the specific ID, among which are the description text and image displayed (at the top of the page), but also a list of keywords related to that specific article. Based on these keywords, a new API call will

be made and from here the first four results are presented as the articles under the "You May Also Be Interested In" heading at the bottom of the page.

Third, I made use of the URLSearchParams API to get query strings from the URL. This enables the website to check whether the user is trying to access the details page or the search results page without any query string, in which case he/she is prompted with a message and returned to the home page. If the URLSearchParams detects a query string, it will cooperate with the fetch API to asynchronously fetch data from the server.

Fourth, a series of array functions have been used. For instance, in the timeline.js, the fetch API will return an array containing all the data related to the Apollo mission for the year 1969. First, I used the sort function to sort the retrieved data by date. However, as there is such a huge number of events from that year, I wanted to limit my results to the period from around April to October so as to avoid an endless timeline. Therefore, I used the slice function to split the sorted array into three different arrays (pre-launch, launching and landing), each containing five elements. I also made use of the forEach function to loop through the three arrays and execute a function on each item.

Fifth, I used regular expression for input /form validation such as checking for a valid email format in the contact.js file.

Last but not least, to manipulate HTML elements and create interactive effects on the client-side, I used a number of DOM and BOM methods:

- the window.location.assign() method was used to modify the URL on the search results page, based on the user's search input;
- the getElementById() method was used to get access to a specific element in the HTML document;

- the addEventListener() method was used to manage mouse clicks and keypress events;
- the hasChildNodes() method was used to conduct a boolean check to check if a parent node has any children;
- the setAttribute() method was used to set an attribute to an element in the HTML file;
- the appendChild() and removeChild() methods were used to add or remove a child node from the parent node.

9. Implementation/rollout

In terms of implementation of the design, I used Trello as my project management tool, where I set up cards under the categories “to do”, “in progress”, “review” and “release”. This made the process trackable, gave me a good overview of how far along I had come, and ensured efficient delivery of tasks.

For prototyping, I used the software Balsamic Mockups 3 and followed a mobile-first approach. The design is adaptive to different devices and screen sizes. For instance, the video on the homepage is hidden on mobile devices, as mobile users usually have limited data available. Furthermore, on the timeline the detailed description of the timeline events will be hidden once the screen is lower than a certain threshold. This means that mobile users will only see the title, the date, the image and the “view more” button, which makes for a more clean and less cluttered mobile-friendly presentation.

For the development process, while writing my HTML and CSS code I made use of a Visual Studio Code extension called Prettier, which automatically creates beautiful and clear code formatting. I also used Git for version control. As shown on my Github repository, the whole development process is trackable. I usually started the development of each feature of the page on a separate branch and then, after its

completion, merged the branch to the master. I also used the git stash command to temporarily save my local changes and then re-apply them later on.

For testing purposes, I made use of several developer tools. I used Postman for testing API queries, Chrome's Developer Tools for testing responsiveness across a number of devices and screen sizes as well as for testing CSS styling and site performance. In addition to these developer tools, several friends and family members have been invited to do manual testing of the microsite functionality, interfaces and responsiveness. I followed a test-driven development approach from early on so as to discover and correct issues as soon as possible.

I also fixed a series of bugs, most of which were connected to the NASA API's default functions. For instance, some of the individual items in the API database don't contain any keywords. So to ensure that the "You may also be interested in" section in the details page always gives suggestions, I wrote a conditional statement to control what will happen when the item does not have keywords. A default search query ("Apollo 11") will be used in these cases, to ensure results. Similarly, in some cases the item has keywords, but when calling the API again based on these, only the current article itself will be returned as a result and thus listed under the "You may also be interested in" section. In these situations too, the default search query of "Apollo 11" will be used, to get different suggested articles.

Furthermore, some items in the API database does not contain any description. Here too, a conditional statement has been used, to inform the user that this item actually does not contain any description. This avoids the user misinterpreting the lack of a description as a website error. Another workaround is connected to the timestamps provided by the API. As a default, the returned date format is a unix timestamp. To reformat and present the date in a more readable manner, I relied on the date method.

The only “bug” I have discovered but not been able to fix, is that the maximum return from a single API call is limited to 100. This is a default for the Rest API. I have searched through the NASA API documentation and around the net, but have not been able to find a workaround to remove this limit for the NASA API. As mentioned, this will be seen by the fact that many search results will be limited to 100. For any searches that does indeed return only 100 or fewer results, however, nothing should be missing (for instance the search result for “Apollo Legacy” is 10, which is correct). Additionally, several of the data from the API contains either the same image (but has a different title and text) or the same title (but has a different image and text), although they really are unique (as seen by their different dates and unique NASA ID). This is not a bug, but may sometimes be a bit confusing, try for instance to search for “Saturn Apollo Program”.

10 Conclusion

Following a mobile-first approach and complying with industry standards in terms of design principles, I am very happy with the end result that has come out of this project. With semantic HTML elements and meta tags the website is optimized for search engines. With aria-attributes, together with a good combination of color and fonts, the accessibility of the page is validated against WCAG requirements. With modern JavaScript syntax I have been able to create sophisticated onsite search functionality as well as make fast API calls that successfully retrieve, interact with and presents the results of these calls in a manner of my choosing. All this together with a good user interface. The end result is a dynamic and responsive website, built from the ground up without the convenience and shortcuts provided by frameworks.

In conclusion, in the last five weeks I have been able to make use of pretty much all the tools I have picked up over the last two semesters, and been forced to pick up some new ones as well. It has been the most challenging project so far by a good margin, yet it has also been the most educational and fun one.