Project Exam 1

Front-End Development 1

By: Ole Alexander Moa Pettersen
Tutor: Eivind Solberg

https://olepettersen.net/Project/index.html

https://github.com/OAMPfed/Project_Exam_1

Table of Content

1. Introduction

This project exam has been an exercise in the student's skill of using basic HTML, CSS, JavaScript, DOM manipulation and design structure without the use of bigger frameworks that ease that burden. It has been the culmination of an entire year's worth of learning coding, design, design philosophies and best practices in relation to the Front-End vocation.

The task at hand was to create a microsite with a minimum of four pages, with additional technical requirements like incorporating DOM manipulation, working with an API and JSON data, focusing on semantic HTML, being fully responsive across mobile, tablet and desktop while also using external files for CSS and JS.

Beyond these requirements the project was very open-ended in the way you could create your own style and focus for a website focusing on either NASA or SpaceX. If possible, you could create a website for a real-life client as well, but the requirements were still there. This leads to the easy decision for me to focus on creating a website for SpaceX which is already a very stylish and technologically interesting company.

2. Planning

The project had a few milestone turn-ins, the first being a turn-in of a functional spec and a Gantt chart.

A Gantt chart is a visual tool to map out all the individual tasks required, a timeline for when these tasks should be done, and by whom the tasks belong to. As a one-man dev team, my Gantt chart was not divided into task owners, but I had colour coded the different weeks into three pieces (planning & research, designing and coding). This gave me a clear oversight of which tasks I had to do and when I had turn-ins or other important dates to pay attention to.

For my functional spec I found a Software Requirements Specification template online which included all possible points that would be relevant to a job that involves creating a product and coding it. I had to strip away quite a few points as they didn't belong within the scope of the project, but the end result was still very specific and detailed.

3. Research

The research was a focus on personas, focus groups and target audience. We also created storyboards in the end that were intended to show a real scenario of people using the website.

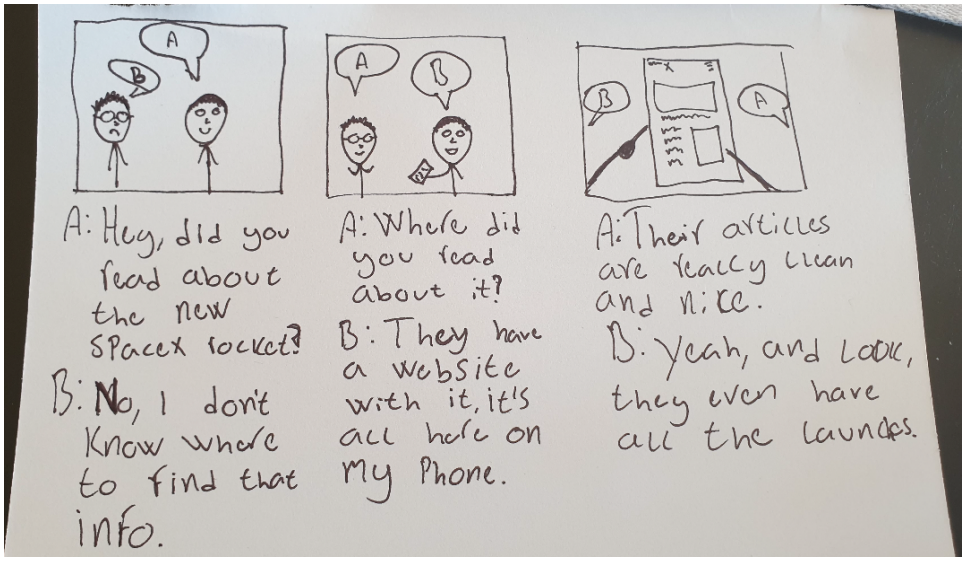

Fig. 1) Storyboard

The personas were of three people I could imagine fitting the target audience. People with an interest in science, sustainability and cutting-edge technology. The target audience is rather specific, but the personas still fit the kind of people that would be interested in learning more about SpaceX and their endeavours.

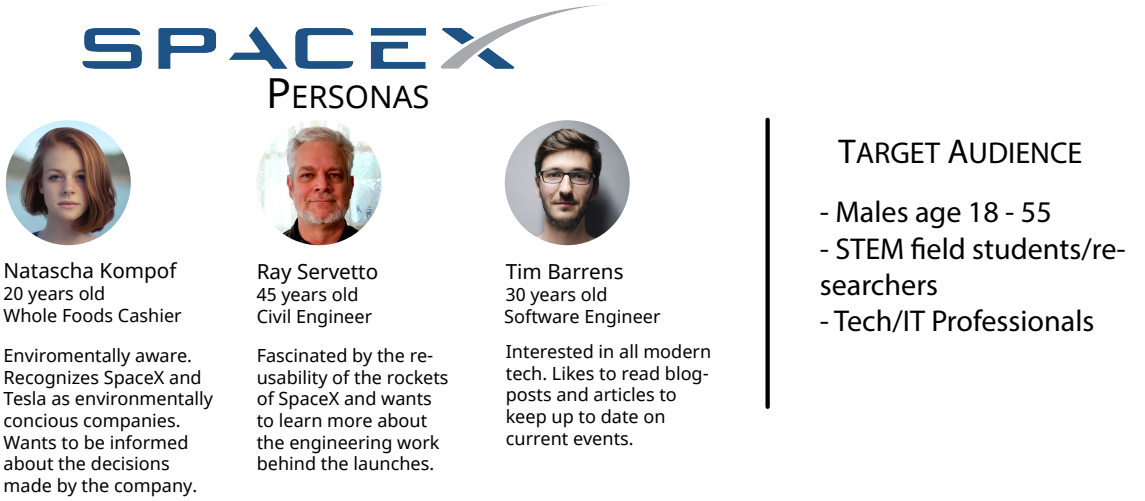

Fig. 2) Personas

The act of researching a target audience was actually much harder than imagined. Despite being closely connected to Tesla and Elon Musk, SpaceX doesn't have a clear target audience lined out online, so I had to rely on a lot of hunches and half-truths presented in various articles and forums online. The personas and target audience fit my personal bias, but this comes from the viewpoint of somebody that actually reads a lot about SpaceX, Tesla and Elon Musk already.

4. CODING

As stated earlier in the report, the assignment called for the use of plain HTML, CSS and JavaScript. Tutor confirmed that students were allowed to use libraries and frameworks for components that fell outside of the immediate scope of the assignment. To keep with the instructions and technical limitations, technologies used were plain JavaScript despite earlier intentions to use libraries like GSAP for animations on SVGs, plain HTML, but SASS was used for CSS. The only benefit of SASS that was used was the ability to nest CSS in the same fashion HTML is nested. More specific information will be found in the subcategory for each technology.

4.1 HTML

I pride myself on focusing on semantic HTML. This helps greatly with SEO and better results from Google's spiders that attempt to distinguish what the structure of a website uses. Running the pages through w3's validator shows only one error which is a divider bar in the link to Google fonts, as this contains two different fonts. It's better to have one additional connection for the fonts, than two. On the launches page an article wrapping a quote is asking for a header tag, but this is nonsensical as the quote and with a separate <p> tag for a name to the bottom right is better wrapped in an <article> tag than in a . wouldn't trigger the validator to use a <header> tag, but it also would be worse semantic HTML.

As such the components of the microsite are never wrapped in s except for the launch cards being used to populate API fetched data, or where needed purely for the sake of positioning components through flexbox. The html tag is after all a block element meant for wrapping content that can't be wrapped by other html tags that would properly describe the content. As an example, the buttons on the 'LAUNCHES' page are wrapped in a as these buttons are meant to be grouped together and as they do not actually lead anywhere, a <nav> tag would be unfit. Therefore a tag to keep them together is used so that styling through CSS, these properly create less columns and more rows on a small screen or more columns and less rows on a bigger screen.

4.2 CSS

As stated, the CSS was written through the use of SASS, or SCSS, as this is a tool that allows for nesting CSS in a logical way similar to HTML. This makes it much easier to respect specificity, as well as the end-result will have all the additional CSS rules for supporting other browsers like -mozkit, or even support for CSS grid on Internet Explorer/Edge. There were very few classes used, in fact the only classes used were those for the navigation menu and for the launch cards. The rest of the microsite has been styled with specific IDs or pseudo-selectors. This might lead to a lot of copying and pasting to have the same style on different components, but due to the logical hierarchy that is followed through nesting, this becomes a very quick endeavour and also allows for easier styling of individual components.

4.3 JavaScript

I am a big believer in “if it can be done without JavaScript, do it without”. This leads to less bloat on a webpage through unnecessary libraries or frameworks used for the sake of keeping things easier. The end result is very often a sluggish site that takes long to load, and this is not good for user retention.

The JavaScript used is to fetch API data and then use specific parts of the JSON data too populate launch cards. These launch cards are created and rendered within the same fetch function that fetched the data. There are also a few variables that function as parameters for the function to only fetch specific data from the API. Initially the plan was to save this data in an array as the fetch was connected to a ‘onload=’ function, but looping through this data later for each different button (that was supposed to only create cards for specific data) over and over would take extra processing power from the browsers, potentially leading to a slow and frustrating experience in the future as the API continues to grow.

The settled upon solution is to fetch fresh data every time a button is clicked, but only the data that fits the parameters, so there is less data overall being fetched. This demands little processing power and only takes half a second at most to populate all the cards for the biggest loads of JSON data. Also if a single object is fetched (like the ‘next’ or ‘latest’ parameter), these are first wrapped in an array so that the function is able to loop through it once and populate that, otherwise the function that renders the cards wouldn’t target this specific data.

The second piece of JavaScript used is to change classes on the navigation menu so that it flows in and out of the website.

5. WCAG AND ACCESSIBILITY

The website is fully WCAG compliant with the exception of the copyrights <p> tag at the bottom of the footer. This is non-important content for a user, and in a real-world scenario it actually doesn’t mean anything within the scope of this project as this is an unofficial project and doesn’t actually represent SpaceX in any capacity. The reason that <p> tag is triggering WCAG rule checks is because it’s a dim grey colour against a white background, which for people with poor eye-sight, colour-blindness or totally impaired eye-sight, the sentences is very hard to see. Due to the nature of the sentence and how un-important it is, the design choice outranks the weight of the WCAG compliance.

There are also issues with the form and how these are structured, but that’s a problem with the HTML structure and not the visuals of the form. A WCAG checker will also state that there are radio buttons required, which do nothing as the form asks for user input through text. Radio buttons do not apply to the form’s intended use.

Over all the feedback from a WCAG and HTML validator are negligible, and I’d make the same decisions again.

6. DESIGN CHOICES AND PROTOTYPING

The project was prototyped in Adobe XD. An excellent program that allows you to create an interactive website once all components are finished and connected. It is in my opinion the superior tool to use for a prototype compared to creating wireframes and such in Adobe Illustrator, although these tools also have specific uses that are important as well.

The end result was slightly different from the prototype as the prototype had to be delivered as a milestone assignment. This meant that newer and better ideas trumped the old ones that were essentially set in stone. The same fonts and colours were still used and the greater design and feel of the site remained the same. The biggest differences were that the navigation bar was changed into a permanent hamburger menu and the crooked lines of the individual sections had to be pushed slightly into the sections as a random 1-pixel line would show between the crooked lines and the square section of content. This leads to a crooked line that ends in the section instead of the far-right of the website, but this is a subtle flaw and is likely never seen by anyone not actively looking for it.

The fonts used were RUSSO ONE and OPEN SANS. Russo One has a futuristic, bold impression and Open Sans is equally futuristic and professional. They are both sans-serif and come across as no-nonsense fonts while also having slightly rounded edges so as to not come across as hostile or edgy. The dark sections of the page follow this same design philosophy by not being completely dark, but slightly grey instead so that they are also less hostile and easier on the eyes. There are lots of stark contrasts, but none that strain the eyes or come across as too strong.

7. CONCLUSION

In conclusion, it feels a lot better to have a four-page minimum compared to the six-page minimum on the semester project. It allows for much more focus on each page and the responsiveness, interactivity and design. As we have learned more and more technologies and have gained a greater understanding of these technologies and the things we are capable of doing with them, it doesn't actually become easier to create a website, it in fact demands more of us. The differences between our first website and this final project are absolutely huge. Not just because we are better at design, but because we gain a deeper understanding of how to better write code and how far we can push the use of these technologies.

I'm very pleased with the end-product and while I realize this is a very amateur website compared to veteran programmers, it's something I'd happily show off and a great exercise for creating future apps or websites this summer before the next semester.