**Design**

**Hardware**

After some rough sketches, I drew up some drafts of some potential designs. At this point I didn’t know the exact dimensions of the mirror as I had not acquired the necessary components yet (LCD, Mirror, etc.), so I chose some arbitrary dimensions in the ratio 16:9 (the most common aspect ratio for TVs and monitors).

I listed some potential solutions, which I will not expand on for brevity.

* Raspberry Pi
* Android TV
* Android tablet
* Google Chromecast
* Amazon Firestick

After considering several designs, I decided that the most convenient way to tackle the “smart” component is via an Android phone which will be beamed to a TV via Chromecast. I feel like this is the most flexible and least restrictive choice. The fact that it works via a phone means that hardware purchases are kept at a minimum. Moreover, instead of using a dedicated controller like a Raspberry Pi, which implies a permanent installation, a streaming device can also be used for more general media purposes, like displaying photos.

Here is a labelled exploded-view diagram of what I visualise to be the final product.

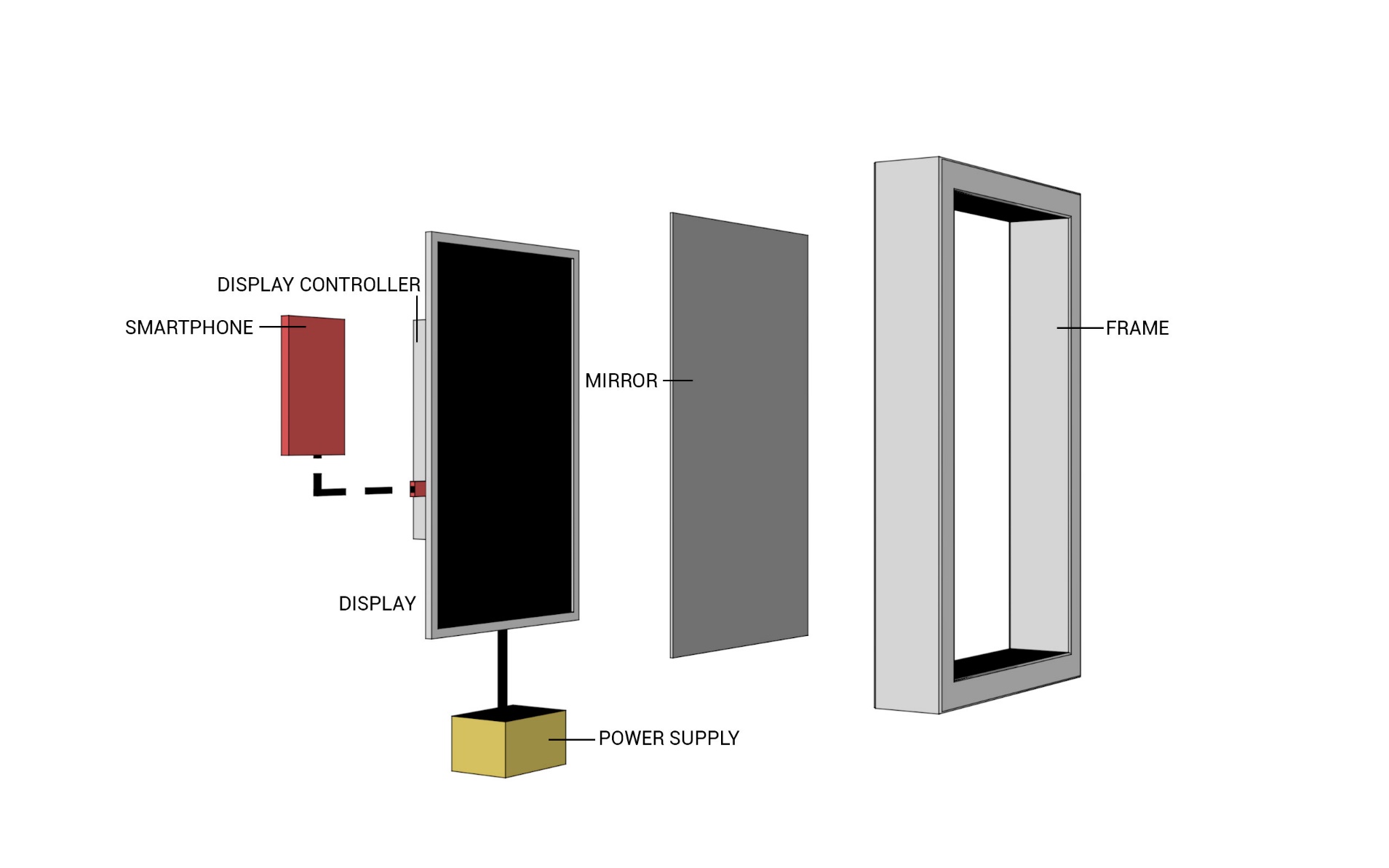


Fig 2.1 – A *“blowout”* view of the construction

**Software**

Initially, when I was first drafting this project I had intended to learn Java and native Android programming. An Android app running on a phone would power the display. However, in February I attended a *Freeformers* hosted “Learn to Code” event where I learned about alternatives to native app development: Mobile and Hybrid apps. These involve using HTML, CSS and JavaScript technologies used for **webpages** instead of Java to create a mobile app. A drawback of mobile apps is that, because they are essentially a website you must be connected to the internet to use them. Hybrid apps solve this by converting a purely mobile app into something that can be installed and used offline. I did some research on the pros and cons of each type and then eliminated those which were not relevant to this particular project[1]. As I am not creating a conventional handheld app and it is a personal project, factors such as the ability to monetise apps by submitting them to an app store or being effective over multiple platforms are not a concern. Essentially, it comes down to if I prefer web development or native development. I found that hybrid app development provided a comfortable compromise between being able to utilise the full capability of the device, the requirement of more general web developer skills, and reduced latency function over HTML5. This fact also means that the app can be used across most platforms, including a browser. The implication being that if you really wanted to use a Raspberry Pi, porting it would require little extra work. Ultimately, I decided that advantages of hybrid development outweighed native development, especially given the timescale and taking into consideration the skills I would develop.

There are definitely some obvious drawbacks with this design which I have considered, such as the fact that the phone must be actively connected to the display, the implication of which is that other apps cannot be used at the same time. However, I don’t see this as a major concern for the reason that this project is primarily a proof-of-concept. Furthermore, hybrid app development means that it can very easily be ported to a platform that uses a dedicated controller i.e. not your phone.

There are a plethora of resources available both online and offline for learning web development. After a significant amount of research I compiled a list of those which I think best suit my needs.

Possibly one of the most popular online resource for learning programming is **Codecademy** [2]. Codecademy is a great, interactive tool to learn programming and provides courses for a large range of programming languages and even an entire section dedicated to web developer skills, encompassing basic HTML to Ruby on Rails web apps. Its interface includes an in-built code editor, and changes are shown in real-time.

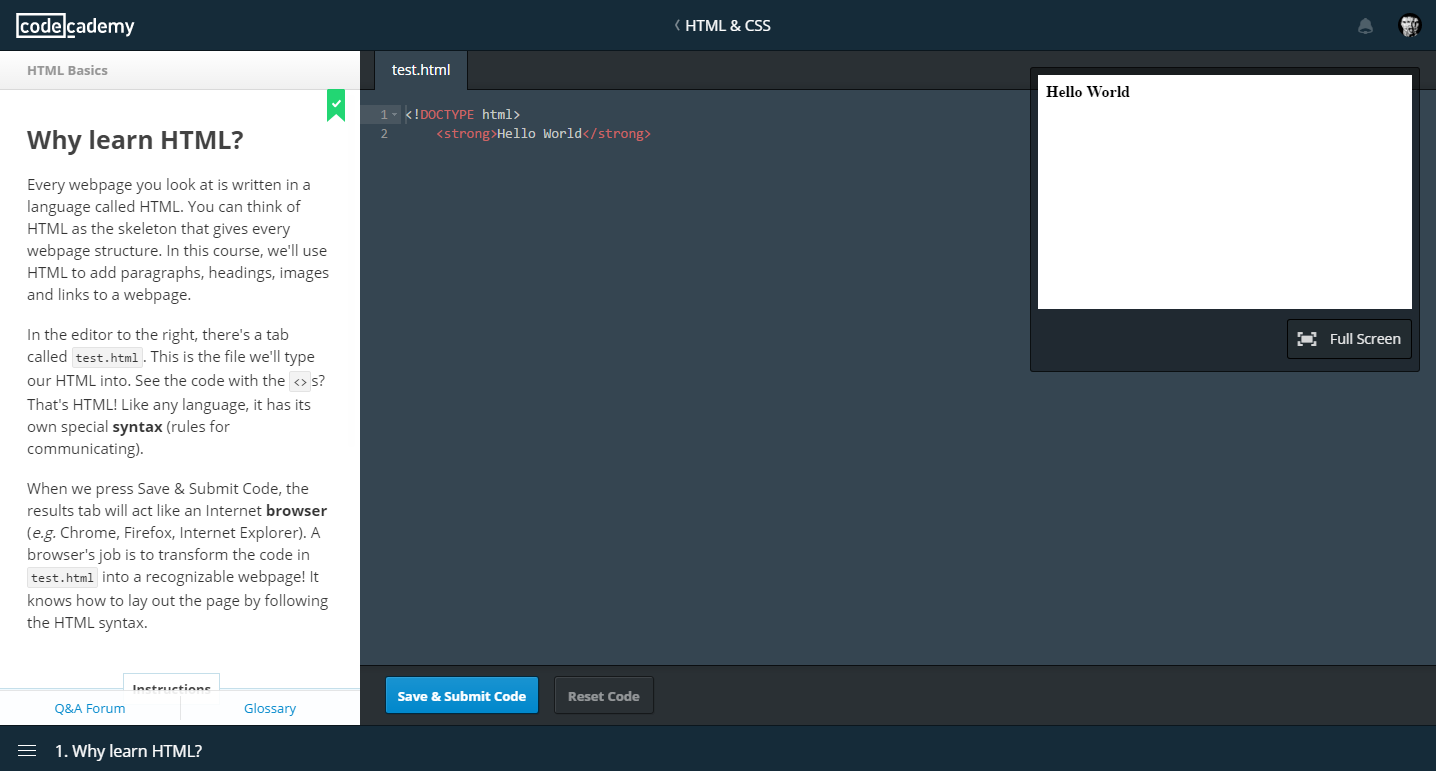
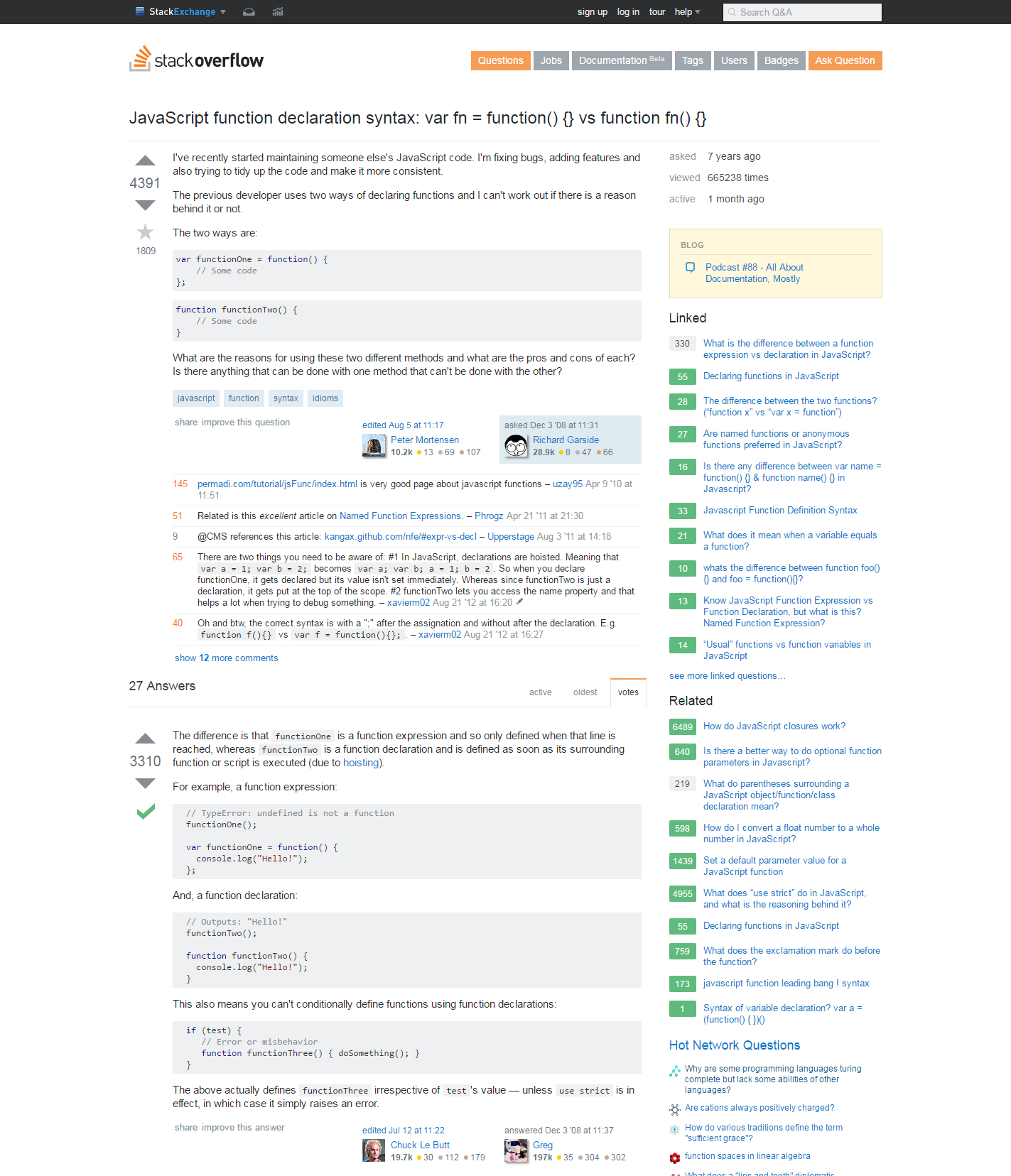


Fig 2.2 – The Codecademy interface

While Codecademy is very useful for getting to grips with the basics, there are many other ways for an aspiring programmer to learn. For one, it would be a cardinal sin in the software development community to neglect to mention **Stack Overflow** [3]. Stack Overflow is an incredibly popular community-moderated question and answer based forum for programmers. The true value of the website comes from the incredible breadth of answers available, and often it seems that every possible niche bug and error has already been covered in detail.



A user makes a post posing a question, including more detailed information about the issue in the body. Other users can “upvote” or “downvote” the submission depending on if it is well-asked (such as being sufficiently non-ambiguous).

Other users can comment on it directly, asking for more detail etc. Or they can post a direct reply, explaining a solution. Again, other users can vote on the quality of the answer. The asker can then mark the response as the best answer, shown by a green tick.

Fig 2.3 – An example of the stack**overflow** layout

I have chosen to not use any books as online media has every advantage against its offline counterpart. The most important instance being the fact that with the speed of development and changes made to the technological world it is inevitable that books will become outdated and of very little use very quickly – not to mention the cost.

**Interface and Icon Design**

While designing this interface I incorporated elements of both Apple’s *Human Interface* [4]and Google’s *Material Design* [5] guidelines. I thought the clean iOS design was a perfect match for the minimalist aesthetic of the mirror. Google’s Roboto font provides a free and open-source alternative to Apple’s San Francisco font, and the use of material design for the icon matches the rest of Google’s unified Android look.

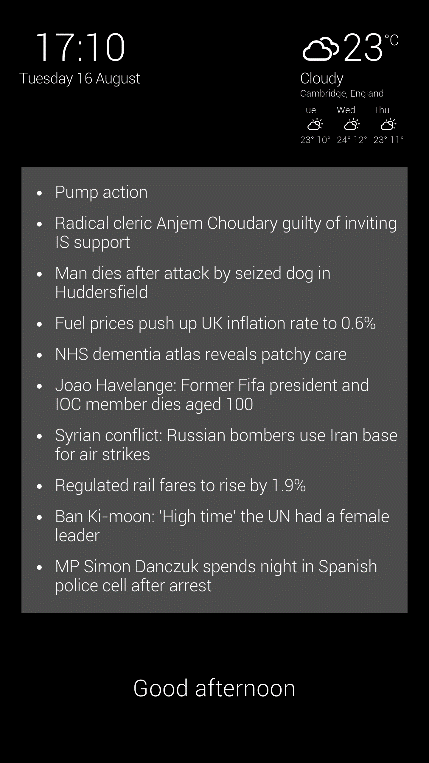
I wanted to provide minimum obscuration of the user’s reflection so I opted to keep the main view mostly empty, and then implement pop-up windows that will come up when the user asks. Here is an example of such a window.

Fig 2.4 – The icon I designed

Fig 2.5

# References

|  |  |
| --- | --- |
| [1] | D. Cortez, "UpTop," 27 August 2015. [Online]. Available: http://www.uptopcorp.com/post/mobile-app-development-native-vs-hybrid-vs-mobile-websites . |
| [2] | "Codecademy," [Online]. Available: https://www.codecademy.com/. |
| [3] | J. Atwood and J. Spolsky, "Stack Overflow," [Online]. Available: http://stackoverflow.com/. |