KTH Computer Science
and Communication

Modular responsive web design

Allowing responsive web modules to respond to custom criterias instead of only viewport size by implementing *element queries*

LUCAS WIENER
lwiener@kth.se

Master's Thesis task description

Introduction

Many good developers agree that modular development is an important factor for successful software development. To break systems up in modules and having these modules responsible of well defined tasks enables programmers to develop their products in a more controlled and reliable way. Solving problems in the context of a module with a single responsibility is simpler than solving problems in a context which does many different things. Furthermore, modules are easy to reuse in different projects due to the single responsibility nature of modules. Again, many developers have seen the benefits of using modules and as of writing this text there are millions of modules (also called plugins, libraries, components, etc.) online for the world to use. Most development environments enable developers to work in a modular way and usually encourage it. Web programmers are used to using third-party libraries and components to build complex web apps with modular interfaces (i.e. views that consists of smaller modular views).

With the rise of smartphones and tablets, web developers realized that they needed a good way of having their websites adapt to the different devices. The solution came to be responsive web design, which is a technique to make websites react and adapt to the size of the viewport (i.e. the browser window), so that the website elements are displayed in a customized way for different viewport sizes. This enables big desktop websites to shrink to smaller and more mobile friendly versions when the browser window is resized. This was a big breakthrough in web development and responsive web design has become very popular (some say it is the de facto technique to solving the content adaption problem). The idea of having components and modules morph to more suitable designs given a size is amazing.

The problem

Imagine a website displaying a news feed. Each post of the news feed contains a lot of information; title, body, location, time, source, author, etc. The app is responsive, so that when the news feed is viewed on a small screen (less than 600 pixels wide for example) only the important information of each news post is shown to the user; title and body. The news feed is a module, so it can easily be integrated into other web apps. But what if another web app just wants to have the smaller version of the news feed? If the web app has the news feed module in a container less than 600 pixels wide the news feed would react to the small width and adapt itself to the smaller version, right? Unfortunately not. This is the issue I would like to address in my master's thesis. Writing responsive modules that can adapt to parent element sizes instead of viewport sizes is a natural step forward in responsive web design. This way web developers would truly be able to build and use modular web components.

Types of solutions

The root of the problem is that responsive web design is heavily dependent on CSS rules to specify different element styles for viewport sizes by using *media query* conditions. By using media queries, elements can be styled in the following manner:

```
@media screen and (max-width: 600px) {
    body {
        background-color: blue;
    }
}

@media screen and (min-width: 601px) {
    body {
        background-color: yellow;
    }
}
```

The above CSS styles the body of the website blue if the viewport is less or equal to 600 pixels wide, and yellow otherwise. So how would one make components react to a parent element size instead of the viewport size? I can think of three approaches:

1. **Wait.** The cleanest way to achieve this would be waiting for browser vendors to implement support for media query-like parent rules. Unfortunately, it is not in the pipeline to be done for any of the major browsers.

2. **Change.** By defining a custom syntax to declare parent response rules one could solve the problem in a fairly clean manner. The major drawback with having custom syntax is that all those millions of responsive web components existing today would need a rewrite in order to work. Also, I believe it would be hard to agree on such syntax design, especially if there are no browser vendors or other organisations supporting it. This approach might be sufficient for companies that just want to create reusable responsive modules internally and are okay with being dependent on a third-party syntax. However, this approach would also require all bits of a module to conform to the syntax. This means that modules and components that are written on top of existing frameworks and libraries (*Bootstrap* and *Foundation* for example) would not work out of the box.

3. **Hack.** By making javascript hacks I believe a solution can be found which enables any media query dependent component react to parent sizes instead of the viewport size. If such solution can be found, all those millions of responsive components could instantaneously be reacting to parent sizes without any of them needing to change. Potential problems with such solution could be that it impacts the average performance in a negative way. Also, support for older web browsers could be troublesome.

Objective

The main objective of my master's thesis is to develop a solution of type 3 in the above list. The scientific question to be answered is if it is possible to solve the problem without extending the current web standrard. The hypothesis is that such a solution can be created which enables existing responsive components to react to a specified criteria (parent container size for example) with no modifications to the components or web standards. By reading CSS specifications, javascript books and by looking at existing open source projects I will try to implement a solution. The goal of the thesis should be considered fulfilled if a solution was successfully implemented or described, or if the problems hindering a solution are thoroughly documented. Not only will such a implementation aid companies building websites, it will also act as a prototype of the idea as a third part plugin, that later can be examined and reviewed by browser vendors when it is time to implement it natively. This way the idea can be tested by early adopters without the community risking to breaking current CSS standards or browsers.

Why me?

My studies at KTH in the field computer science have made me a decent programmer and I have read numerous web related courses. While studying at KTH, I have been running a web development company which has created web applications, plugins and solutions for other web related problems. Back in the days when *Facebook* still was something new, me and my colleague created a web chat based on a custom streaming method to deliver chat messages to clients in real time. The streaming was implemented with a long-polling technique. At this time most of the web based chats (the Facebook chat included) used a polling technique that fetched new messages in a static interval. One year later, Facebook implemented a long-polling technique similar to ours and later streaming was widely used by using the freshly browser implemented web sockets. With this said, I love pushing the web forward and I believe I have proven myself capable of doing so.

Resources

The subject has been brought to my attention by Tomas Ekholm (client-side web architect at EVRY AB), and will together with his colleague Stefan Sennero (responsible of the area of competence at EVRY AB) assist me to solve the problem. Tomas is also an associate professor in mathematics, and teaches programming patterns (among other courses) at KTH. No earlier research or work has been done regarding the problem. Since both Tomas and Stefan are very competent and technically skilled in the field they will be a huge resource in developing a solution.

The work will mainly take place at EVRY AB so that we can have a continous dialog regarding the work. Also, by sharing workspace I will be able to fully utilize the compentence of Tomas and Stefan.

Both parts would like the solution to be open sourced, which could potentially lead to external feedback and help from other developers in need of the solution. This also enables me to more freely use existing solutions to the subproblems that I will face.