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 that 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 (it is arguably 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.

Background

As already stated, CSS today enable developers to make elements adapt to the viewport sizes through *media queries*. The queries are designed in a way that one can define element style rules that will be applied if the viewport criterias are met. 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;
    }
}
```

Listing 1. The above CSS styles the body of the website blue if the viewport is less or equal to 600 pixels wide, and yellow otherwise.

The first public working draft for media queries was published in 2001. In 2012 W3C[1] made media queries an official recommendation. Developers quickly realized the need for media queries that works for elements instead of viewport size, which came to be known as *element queries*. Around 2013, the discussion about element queries blossomed up and became a hot topic for web developers. People wrote articles about the problem, some provided proof of concept implementations and some emailed W3C to discuss potential native support in CSS. W3C responded that such feature would be infeasible to implement. Some problems with implementing element queries in CSS are:

- **Circularity:** The styling of elements depend on many factors (theoretically on all other elements in the tree). If elements can apply styles by criterias of other elements, it will be possible to create infinite loops of styling. The simplest example of this would be an element to set its width to 200 pixels if it is under 100 pixels wide. If the element is under 100 pixels wide, the new style will be applied to the element which would make the width of the element 200 pixels. If this element would have another rule that set its width to 50 pixels if it is wider than 150 pixels, there is an infinite loop of styling. Problems like this can probably be catched during CSS parsing, but there are so many combinations of style properties that could result in similar loops that it will add a lot of complexity to the language, both for implementers and users.

[1] World Wide Web Consortium (abbreviated W3C) is the main international standards organization for the World Wide Web.

- **Performance:** Rendering engines typically perform selector matching and layout computations in parallel to achieve good performance. If element queries would be implemented, the rendering engines would need to first compute the layout of all elements in order to decide which selectors would conform to the element query conditions and then do a new layout computation, and so on until a stable state has been reached. Far worse, since selectors now depend on layout style, this cannot be done in parallel which impacts performance heavily.

Because of the problems, it is stated that such feature will not be implemented in the near future. So it is now up to the developers to implement this feature as a third-party solution. Efforts have been made by big players to create a robust implementation, with moderate success. Since all implementations have shortcomings, there is still no de facto solution that developers use.

Objective

The main objective of my master's thesis is to develop a third-party implementation of element queries (or equivalent to solve the problem of modular responsive elements). To do this, I will need to research and understand all existing attempts and analyze the advantages and shortcomings of each type of implementation. I will also need to be aware of the premises, such as browser limitations and specifications that need to be conformed. There are many challenges along the way that will need to be researched and worked around. Examples of such subproblems that would need to be investigated are:

- How should circularity be handled? Should it be detected at runtime or parse time, and what should happen on detection?
- How can one listen to element dimension changes with javascript without any native support?
- Can custom viewport elements such as *object* or *svg* be utilized to make regular media queries perform with respect to the elements?
- How can a custom API be crafted that will enable element queries and still conform to the CSS specification?
- If a custom API is developed, how would one make third-party modules (that uses media queries) work without demanding a rewrite of all third-party modules?

The scientific question to be answered is if it is possible to solve the problem without extending the current web standard. The hypothesis is that the problem can be solved in a reliable and performant way by crafting a third-party implementation. A reliable implementation should also enable existing responsive components to react

to a specified criteria (parent container size for example) with no modifications to the components. 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.

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. Since both Tomas and Stefan are very competent and technically skilled in the field they will be a huge resource in developing a solution.

I will base my work on existing projects and implementations. There are many articles on the internet regarding the problem and potential solutions. The W3C public mailing list contains valuable information regarding problems with a native implementation as well as insight into how a third-party solution could look like. I will also read books and other research papers in order to solve the subproblems that I will be faced with.

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.