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 specifikation

Supervisors at EVRY AB: Tomas Ekholm & Stefan Sennerö
Supervisor at CSC: Philipp Haller
Examiner: Mads Dam

Problem definition

By using CSS *media queries* developers can specify different rules for different viewport sizes. This is fundamental to create responsive web applications. If developers want to build modular applications by composing the application by smaller components (elements, scripts, styles, etc.) the media queries are no longer applicable. Modular components should be able to react and change style depending on the given size that the component has been given by the application, not viewport size. The problem can be formulated as: *Elements can not specify style rules depending on their own size, or the size of any other element.*

W3C[1] has unofficially stated 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.

- **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 and the problem remains unsolved.

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

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 approach. I will also need to be aware of the premises, such as browser limitations and specifications that need to be conformed. In addition, I will research the problems of implementing element queries natively to get a deeper understanding of how an official API would look like. 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 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.

Significance

Web developers are today limited to writing big applications in an entagled mess. In almost all other programming environments it is possible and encouraged to write applications in smaller parts (or modules). By creating modules that can be used in any context with well defined responsibilites and dependencies, developing applications is reduced to the task of simply configuring modules (to some extent). It is today possible to write the web client logic in a modular way in javascript. The

desire of writing modular code can be shown by the popularity of frameworks that helps dividing up the client code into modules. The ever so popular frameworks Angular, Backbone, Ember, Web Components, requirejs, Browserify, Polymer, React and many more all have in common that they embrace coding modular components. Many of these frameworks also helps with dividing the HTML up into modules, creating small packages of style, markup and code. One of the biggest issues keeping the modules from being truly modular is that they cannot adapt to given sizes. This makes the modules either force the client to style them properly depending on viewport size, or not being responsive. Both options are undesirable for developing larger applications. A third option would be to make the modules context aware and style themselves according to the viewport, which defeats the purpose of modules (making them not reusable).

The last couple of years a lot of articles have been written about the problem and how badly we need element queries. As already stated, third-party implementation efforts have been made by small and big players, with moderate success. W3C keep getting requests and questions about it, but the answer seems to lean towards no. An organization called Responsive Issues Community Group (RICG) have started an initial planning regarding element queries. However, things are moving slow and a draft about element queries use cases are still being made.

Solving this problem would push to web forward a big leap, enabling developers to create truly modular components. By studying the problem, identifying approaches and providing a third-party solution the community can take a step closer to solve the problem. If the hypothesis holds, developers will be able to use element queries in the near future, while waiting for W3C to make their verdict. The outcome of this thesis can also be helpful for W3C and others to get an overview of the problem and possibly get ideas how subproblems can be handled.