Allowing Responsive Web Modules

1st Author Name
Affiliation
City, Country
e-mail address

2nd Author Name
Affiliation
City, Country
e-mail address

3rd Author Name
Affiliation
City, Country
e-mail address

ABSTRACT

UPDATED—September 14, 2015. This sample paper describes the formatting requirements for SIGCHI conference proceedings, and offers recommendations on writing for the worldwide SIGCHI readership. Please review this document even if you have submitted to SIGCHI conferences before, as some format details have changed relative to previous years. Abstracts should be about 150 words and are required.

Author Keywords

Authors' choice; of terms; separated; by semicolons; commas, within terms only; this section is required.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; See http://acm.org/about/class/1998/ for the full list of ACM classifiers. This section is required.

INTRODUCTION

- Why modules? Reusability (even across applications), reduced code complexity.
- Why responsive design?
- Responsive Modules of today need to be context aware (thus, not very reusable [they only work in a specific layout]).
- What do we want and why? Modules that are responsive relative to its outer frame.

A module is an interchangeable and independent part of a program that typically has a single and well-defined responsibility. Modular programming is a technique to reduce complexity and enable reusability. In order for a module to be reusable it must not assume in which context it is being used.

Responsive Web Design (RWD) is an approach to make the application design respond to the viewport size, in order to support varying devices. This is achieved by using CSS media queries to define conditional style rules.

The problem is that there is no way to make a module responsive without it being context-aware, due to media queries only being able to target the viewport. Thus, a responsive module using element queries is layout dependent and has therefore limited reusability.

The desired behavior of a responsive module is having its inner design responding to the size of *its frame* instead of the viewport. Only then is a responsive module independent of its layout context.

This can be achieved with the theoretical feature *element queries* that enables conditional CSS rules by an arbitrary element size. This note presents a novel implementation of element queries in JavaScript, and discusses the new possibilities of GUI design.

EXAMPLES OF BROKEN RWD TODAY

- MQ is not the solution to RWD. (MQ was not designed for RWD as the feature was released long before RWD)
- All elements adapt their inner design by the viewport width.
- Menu Example shows how MQ are broken.
- Limitations of MQ regarding font-size (em).

A SOLUTION

- Parents should decide the layout of their children, and the children should adapt their inner design accordingly.
- Valid language syntaxes (HTML, CSS, JS).

WHY IS A NATIVE IMPLEMENTATION TROUBLESOME?

- Performance issues.
- Cite Tab Atkins of RICG (he states that it is infeasible to standardize this).

A JAVASCRIPT IMPLEMENTATION

- Why is this pragmatic? Compatability, no impact (performance, language) on apps that do not need responsive modules.
- Satisfies the requirements for a solution given above.
- Present Elq's API.
- Present the performance.
- Note drawbacks (but only drawbacks for added functionality!).

DISCUSSION AND SUMMARY OF RELATED WORK

- Performance, APIs, Features.
- The mirror functionality of Elq makes it uniquely suitable for nested modules.

CONCLUSION

- Production ready.
- Probably no standard (or not in a long time).

ACKNOWLEDGMENTS

Sample text: We thank all the volunteers, and all publications support and staff, who wrote and provided helpful comments on previous versions of this document. Authors 1, 2, and 3 gratefully acknowledge the grant from NSF (#1234–2012–ABC). *This whole paragraph is just an example.*

REFERENCES