On the Design of a Responsive User Interface for a Multi-Device Web Service

Jari-Pekka Voutilainen and Jaakko Salonen
Gofore Ltd. Tampere, Finland
Email: jari.voutilainen@iki.fi, jaakko.salonen@iki.fi

Tommi Mikkonen
Tampere University of Technology, Tampere, Finland
Email: tommi.mikkonen@tut.fi

***Abstract*—The introduction of numerous new, web-enabled devices is leading to a change in the mindset in the design of web systems. Increasingly often, they are designed to take into account the special characteristics of different devices. In terms of implementation, designing these responsive web services requires frameworks and guidelines for user interface design for both desktop and mobile devices. In this paper we present a case study which was done during the implementation of opendata.fi service. During the project responsive mobile user interfaces were implemented to existing open source software with preserving as much of the original layouts as possible. We provide some insights to tools and libraries used in building the responsive user interfaces and share some of the findings that work well in customizing existing software. We also provide examples of solutions that wont work so well and look how to solve them properly.**

Keywords: Responsive web design, multi-device software, web development, mobile apps.

I. Introduction

In the past ten years, PCs and smartphones have evolved from high-tech products to consumer commodities. Today, the average U.S. or Europe consumer has at least two primary computing devices: a personal computer and a smartphone. We currently are at tipping point with connected devices, entering a new era of multiple device ownership, where device interoperability is expected as default.[1]

In the past, web applications had and still have separate mobile user interface which may or may not have the same features as the full desktop UI. For example see https://twitter.com/ and https://mobile.twitter.com/ where mobile application has fewer features. Jody Resnick said that the mobile version of the web site is not good enough anymore in terms of marketing, search engine optimization (SEO) and that responsive design makes it easier to manage the content since there is only one website to update.[2] In responsive web design, web sites are designed to take account the different types of devices that the users use to access the web site. This has been extensively researched, and corresponding design practices for building better web sites are well known. However, in many cases we have legacy applications, which have been implemented without thinking about user interfaces other than the desktop. Transforming these applications to modern responsive UIs is not a trivial task.

In this paper we present the findings from the implementation of opendata.fi, a web site for storing, sharing and managing open data sets of different types. The goal of the project was to implement responsive mobile user interfaces for different features with minimal modifications to original templates to support software upgrades to the platform in the future. We developed numerous implementations iteratively, and designed build processes to support large modifications that require minimal effort.

II. Background

A. Multi-Device Development

Today, the average consumer in the U.S. or Europe has two primary computing devices: a personal computer (usually a laptop) and a smartphone. Every day, over 3.5 million new mobile devices and tablets are activated worldwide We are quickly moving from a world in which each person has two or three devices – a PC, smartphone and tablet – to a world in which people will use dozens of devices in their daily lives: laptops, phones, tablets and ”phablets” of various sizes, game consoles, TVs, car displays, digital photo frames, home appliances, and so on – all of them connected to the Internet. Increasingly, the expectation is that the users shall be able to effortlessly roam between all the computing devices that they happen to have.

Creating applications that are well-suited for all of them is the next software engineering challenge for web and mobile software. Applications must be designed such that the user's applications and data shall be synchronized transparently between all the computing devices that the user has, as long as the application and data make sense for each device. Moreover, whenever applicable, roaming between multiple devices shall include the synchronization of the full state of each application, so that the users can seamlessly continue their previous activities on any device.

In addition to delivering a technical solution in the form of a framework, we also need tools and techniques for creating adaptations between different devices with different form factors, user interfaces, and modalities. Creating applications that are able to adapt to such different devices (see Figure 1 [1] for some common categories) then calls for considerations regarding the different alternatives during the design.

B. Responsive Web Design

The goal of responsive web design [3] is the creation of web sites that take into account different types of devices, usually ranging from mobile phones to desktops, and optimize viewing

Fig. 1. Different screen types that must be considered in future web application development.

experience for the device at hand. In particular, aspects such as easy reading and navigation with a minimum of resizing, panning, and scrolling are usually considered important. A key issue in responsive design is how to design the layout and the elements so that they can easily be customised for different screen sizes. Usual implementation techniques include using proportion-based grids and flexible images, where element sizing takes place using relative units instead of absolute ones, and CSS3 media queries, where different styles can be used for different devices.

In general, due to the differences in browsers used in desktops and mobile devices, the development of responsive web designs calls for architecture where some of the functionalities are always executed the server side. This simplifies the development, as these functionalities need not be tested with every possible client device. Moreover, their adaptations for different screen sizes and device types need not be considered, apart from their eventual visual appearance on screen.

Again, numerous libraries exist for building responsive designs. These include Bootstrap[1] and Foundation[2], to name some commonly used systems.

III. Problem Setting

Opendata.fi[3] is a service to Finnish national open data and tools, which provides descriptions and availability information – metadata – about open data in Finland. The objective of the service is to promote findability and availability of open public data, and enable the effortless utilization and adoption of open data in different applications.

This research stems from the need to extend the platform that powers Opendata.fi – which has been in development for the past 9 years, while mobile UIs have been largely ignored – with fast, mobile optimized user interfaces. The easiest solution would have been just redesign all the templates and be done with it, but we wanted to preserve the original templates to the largest possible extent due to the eventual platform upgrades. These upgrades will modify the original templates, and we were looking ahead to incorporate these modifications with minimal manual work.

There were multiple external requirements for the project, ranging from client needs to UX designers layout designs and technical limitations of the Open Source software platforms. The research approach was Action Design Research (ADR)[4], where we develop initial implementation and iteratively improved it based on the the feedback from the internal and external sources, where internal are the developer team and external, for instance, project client, governmental management, and open data activists who use the service.

IV. Design and Implementation

The design and implementation presented here is mainly based to the technical requirements of the project. However, Open Source software platforms were chosen based on a preliminary study executed by an external software consulting company regarding how these kinds of services are usually implemented in different countries.

A. Background

The Opendata.fi service is built upon two open source projects – Drupal[4] and CKAN[5]. Drupal is used as a content management system (CMS) platform for news, blog posts and other similar things, where CKAN is used as basis for a metadata catalogue. From the architectural perspective, these services are installed side-by-side with the Nginx web server acting as a proxy in front to the users. The architecture is presented in Figure 2.

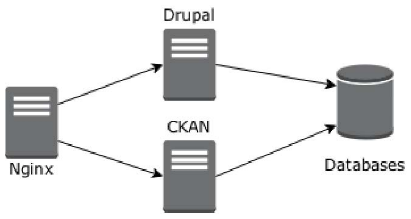

Fig. 2. General architecture of the web service.

Drupal is an Open Source content management system written using PHP. It is distributed under the GNU General Public License (GPL). Individuals, teams, and communities leverage Drupal's features to easily publish, manage and organize content on variety of websites, ranging from personal blogs to large corporate and government sites. Drupal was designed to be enhanced and extended with new features and custom behavior by using add-on modules. There are thousands of such additional modules that extend Drupal's core functionality, covering a broad spectrum of capabilities [5]. There are currently 960 000 web sites running core Drupal 7.[6]

CKAN is a powerful data management system that makes data accessible. The system provides tools to streamline publishing, sharing, finding, and using data. The development of CKAN is managed by Open Knowledge Foundation. Released under an Open Source licence, CKAN is free to download and

[1]http://getbootstrap.com/
[2]http://foundation.zurb.com/
[3]https://www.opendata.fi/en

[4]https://www.drupal.org/
[5]http://ckan.org/
[6]https://www.drupal.org/project/usage/drupal

install for any kind of use. CKAN is in use by numerous organizations ranging from communities and academia to regional and national governments.[7]

B. Enabling responsive layouts

There are multiple options for enabling use of responsive layouts in web applications. The developer can implement everything from scratch or take advantage from the many UI libraries that offer responsive tools. With CKAN, the decision is straightforward, as it uses Bootstrap for UI elements, making it natural to implement the whole service with the same library.

The Bootstrap system provides HTML and CSS based design templates for layouts, typography, forms, buttons, navigation, and other user interface components. These components can be customized with CSS and optionally with JavaScript. In 20th of August 2013 Bootstrap version 3.0 was released. This version follows the "mobile first" ideology from the beginning, and mobile styles are found throughout the entire library.

Bootstrap layouts are based on a grid system. By default Bootstrap defines 12 columns within a single row. These 12 columns are always fitted within the parent row element, disregarding the actual size. With CSS classes, the developer can define elements that span over multiple columns as long as the sum of columns does not exceed 12. Some examples are presented in Figure 3.

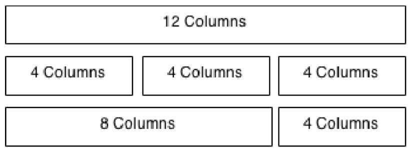

Fig. 3. Bootstrap grid column examples.

In addition to its flexible grid system, Bootstrap also provides utilities for developing responsive layouts. These utilities enable different layouts for different screen resolutions. Utilities are implemented with CSS media queries[6]. These queries are used to replace CSS attributes based on the actual device that is used to render the web page. In Bootstrap's case, this is decided based on the width of the browser window.

Due to the long history of CKAN platform, its web UIs were implemented using Bootstrap 2 which had its final release on 27th of July 2013. Since our Drupal site was built using Bootstrap 3, we decided to also migrate CKAN's theme to version 3. This migration was mainly implemented by replicating CSS classes in sections that were changed. In some cases, modifications were needed for templates. However most of the problems were easily solved by using Bootstrap migration guide[8].

Our whole frontend CSS theme consists of multiple LESS files which are CSS pre-processor language files that extend CSS with variables, mixins, function and many other features.[9] These files are built into single CSS file using gulp[10] and copied into resource directory along with the JavaScript and image files. Gulp is a streaming build system where the developer defines tasks, and gulp executes them in parallel using Node.js streams so that intermediary files are never written to the hard drive. Our gulp tasks consists of modifying Bootstrap and CKAN colors, compiling single CSS of our own LESS files, copying and minifying images, vendor JavaScript and CSS files to correct locations, and finally static error pages are generated. The build process is presented in Figure 4.

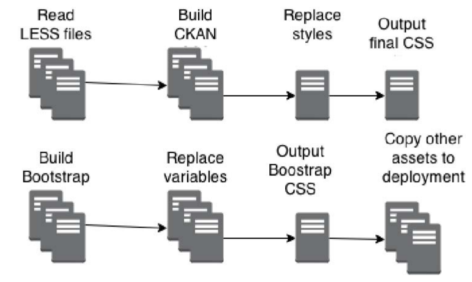

Fig. 4. Build process of frontend with gulp.

C. Implementing mobile UIs as an extension

CKAN is built upon Pylons[11], which uses Jinja[12] as templating engine. Similar conventions to layouts are also found in other template engines, for example in Jade[13] for Node.js. CKAN itself is designed with customizations in mind. Using Jinja conventions, extensions to CKAN can override basic templates. Jinja defines layouts using system called blocks, where inherited templates can override parent blocks. If the extension template has a block with the same name, the template overrides the parent block. However, the parent implementation can be used by calling super() in the extended block. Consequently, by extending the template, the developer can add HTML elements around the original template elements or completely replace them. In cases like reordering HTML elements, the overriding functionality is used in what essentially is copy-paste code from the original template. The easiest cases in customizing the templates can done by only modifying the CSS or by adding new CSS classes to the template.

To implement mobile UIs to existing templates, the templates must first be made responsive. In most cases, it is enough to wrap the existing template blocks to grid defining elements of Bootstrap. For example wrap template block with Bootstrap row and the block elements are automatically fitted to the size of the browser window. To build useful mobile UIs, UI elements must be large enough on the mobile screen. This can be achieved with responsive helpers, where elements in a full size browser can be placed side-by-side, but with smaller screens are placed top of each other. An example of such design is provided in Figure 5.

[7]http://ckan.org/instances
[8]http://getbootstrap.com/migration/
[9]http://lesscss.org/
[10]http://gulpjs.com/
[11]http://www.pylonsproject.org/projects/pylons-framework/about
[12]http://jinja.pocoo.org/
[13]http://jade-lang.com/

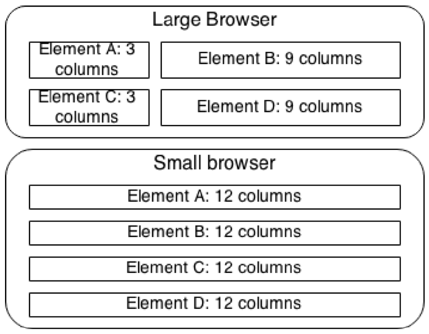

Fig. 5. Example of two browser sizes, where layout is reoriented with responsive helpers.

The main content is optimized in mobile UIs, so it is always first to be seen in small screen. However, since the main navigation options are required to be seen without searching for them, they cannot be placed above or below the main content, because in this case the main content or navigations will not fit the screen. Due to this, the navigation is placed off screen with a toggle button that slides the content to the left and the right. This functionality is presented in Figure 6.

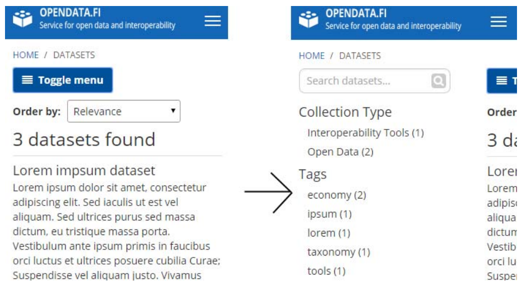
OPENDATA.FI
Service for open data and interoperability
HOME / DATASETS
Toggle menu
Order by: Relevance
3 datasets found
Lorem impsum dataset
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed iaculis ut est vel aliquam. Sed ultrices purus sed massa dictum, eu tristique massa porta. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Suspendisse vel aliquam justo. Vivamus
OPENDATA.FI
Service for open data and interoperability
HOME / DATASETS
Search datasets...
Collection Type
Interoperability Tools (1)
Open Data (2)
Tags
economy (2)
ipsum (1)
lorem (1)
taxonomy (1)
tools (1)

Fig. 6. Mobile off screen navigation.

V. Lessons learned

During the development, some issues with the implementation were discovered. Originally we used pre-built CKAN CSS files and replaced various styles with our own implementations. This affected negatively the Drupal side of the application, and a better solution was to build single CSS for both sides from the ground up with original LESS files within the build pipeline.

Single CSS for Drupal and CKAN sometimes creates issues due to resulting overly general CSS styles. For example if we wish to make CKAN headers bigger, the change might be reflected to Drupal side too, which sometimes is not noticed until later on. Different CSS files for each side with shared styles embedded during build pipeline would solve this but as for now it has not been implemented yet.

In most cases CSS style modifications are enough to redesign the layout of the user interface. With CSS we can move elements around the layout, but it has its limitations. Since CSS was not designed to reorder elements in vertical order, it is relatively hard to achieve this. In these cases we had to copy-paste the original template to our extension with the only difference being in the order of elements. This solution is not pretty one and in some cases it would have been better to just re-implement the whole template.

Our findings are reproducerable in other web applications which use similar block based template engines. These include Jade, Swig, Django templates and Nunjucks among others. All of these are designed for extending through inheritance.

Using gulp as a build system has given great benefits for work efficiency. CSS styles can be modularized so that styles affecting certain elements can be easily found from LESS files. For example styles for navigation elements can be in one file and styles for primary content in other. Gulp also makes it easy to change color schemes and other basic variables like margins since gulp builds custom Bootstrap during the pipeline.

There have been similar projects both in academia and in industry. Data.gov.uk, which is a similar CKAN instance of national open data, has implemented responsive user interfaces for CKAN. However, they have built the templates from scratch, so UI has similar features but with very different implementation. Bernstein and Klemmer did similar redesign to existing web site, but they did it programmatically with their system called ReMorph[7].

VI. Conclusions

The design of responsive mobile user interfaces should be a high-priority task, and this need should be recognized early on in the development of web applications. ”Mobile first” UI design promotes multi-device interfaces as larger UIs are easier to implement afterwards than smaller ones. However there will be cases where the developer does not have an option to redesign the whole UI from scratch to suit mobile needs. Then, modern tools and libraries can overcome problems related to implementing responsive layouts. Our findings indicate that it is possible to make existing web applications responsive with minimal manual work. However, there are some situations that cannot be solved with just using libraries, but such need to be identified during the implementation, with optional settling to non-optimal solutions.

References

[1] A. Taivalsaari, T. Mikkonen, and K. Systä, “Liquid software manifesto: The era of multiple device ownership and its implications for software architecture,” in *Proceedings of the 38th Annual IEEE International Computers, Software, and Applications Conference*. IEEE Computer Society, Jul. 2014.
[2] S. Gunelius, “Why you need to prioritize responsive design right now,” http://www.forbes.com/sites/work-in-progress/2013/03/26/why-you-need-to-prioritize-responsive-design-right-now/, 2013.
[3] E. Marcotte, *Responsive web design*. Editions Eyrolles, 2011.
[4] M. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, “Action design research,” *MIS Quarterly*, 2011.
[5] T. Tomlinson, *Beginning Drupal 7*. Apress, 2010.
[6] H. W. Lie, T. Celik, D. Glazman, and A. van Kesteren, “Media queries,” *Work in progress. W3C Working Drafts are available at http://www. w3. org/TR*, 2012.
[7] G. L. Bernstein and S. Klemmer, “Towards responsive retargeting of existing websites,” in *Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology*. ACM, 2014, pp. 119–120.